Цифровая обработка сигналов

Лабораторная работа № 7

Линейная фильтрация

Содержание

Организация:	Самарский университет
Подразделение:	Кафедра геоинформатики и информационной безопасности
Версия:	2023.11.27-rc

1 Теоретические сведения

1.1 Квазиоптимальная фильтрация

Линейная модель наблюдения в дискретном времени:

$$y[n] = x[n] * h[n] + v[n],$$

где $y[n]$ — наблюдаемый сигнал, $x[n]$ — полезный сигнал, $h[n]$ — импульсная характеристика искажающей системы, $v[n]$ — шум.

Восстановление полезного сигнала осуществляется с помощью ЛИС-системы с импульсной характеристикой $h_{\text{В}}[n]$:

$$\widehat{x}[n] = y[n] * h_{\text{В}}[n].$$

Если восстанавливающий фильтр строится так, чтобы обеспечить минимум среднеквадратичной ошибки восстановления, то его отсчёты определяются уравнением Винера-Хопфа:

$$\begin{cases} \sum\limits_{k \in D} h_{\text{В}}[k] R_y[m-k] = R_{xy}[-m], & m \in D, \\ h_{\text{В}}[m] = 0, & m \notin D, \end{cases}$$

где D — область определения восстанавливающего фильтра.

Средняя ошибка восстановления может быть оценена по формуле:

$$\sigma_e^2 = \sigma_x^2 - \sum_{m \in D} h_{\text{В}}[m] R_{xy}[-m].$$

Если сигнал и шум независимы ($R_{xv}[n] = 0$), то $R_{xy}[n]$ и $R_y[n]$ выражаются следующим образом:

$$\begin{aligned} R_{xy}[n] &= R_x[n] * h[n], \\ R_y[n] &= R_x[n] * h[n] * h[-n] + R_v[n]. \end{aligned}$$

1.2 Развёртка сигнала

Развёртка (обратная свёртка, деконволюция) — операция, обратная свёртке сигналов. Пусть $g[n] = f[n] * h[n]$. Задача заключается в восстановлении сигнала $f[n]$ по имеющимся сигналам $g[n]$ и $h[n]$.

По теореме о свёртке $G(z) = F(z) \cdot H(z)$. Отсюда $F(z) = G(z) \cdot \dfrac{1}{H(z)}$, то есть для получения сигнала $f[n]$ необходимо к сигналу $g[n]$ применить фильтр с передаточной функцией вида $\dfrac{1}{H(z)}$.

Пусть $h[n]$ — конечная импульсная характеристика, состоящая из N отсчётов. Тогда передаточная функция $H(z) = \sum_{n=0}^{N-1} h[n]z^{-n}$ имеет вид многочлена, который можно факторизовать, то есть представить в виде $h[0] \cdot \prod_{k=0}^{N-1}(1 - p_k z^{-1})$, где p_k — корни многочлена $H(z)$. В итоге передаточная функция для операции развёртки примет вид

$$\frac{1}{h[0] \cdot \prod_{k=0}^{N-1}(1 - p_k z^{-1})}, \ |z| > \max_k |p_k|.$$

Для получения адекватных результатов данная система должна быть устойчивой, то есть $\max_k |p_k| < 1$.

В практическом плане для вычисления развёртки может использоваться `scipy.signal.deconvolve`, который внутренне реализован через деление многочленов (с учётом конечности всех сигналов). Обратите внимание, что функция возвращает два сигнала — результат развёртки и остаток. **При применении этой функции необходимо убедиться в том, что процедура восстановления будет устойчивой, в противном случае на результат её работы полагаться нельзя.** Сделать это можно явно (определить максимальный модуль корня многочлена $\max_k |p_k|$ и убедиться, что он меньше 1) или косвенно (значения остатка должны быть близки к нулю).

Рассмотрим случай, когда надо произвести развёртку для уравнения вида $g[n] = f[n] * h[n] * h[-n]$. В этом случае развёртка задаётся передаточной функцией $\frac{1}{H(z) \cdot H(z^{-1})}$. Допустим, что передаточная функция $\frac{1}{H(z)}$ удовлетворяет условию устойчивости: $\max_k |p_k| < 1$. Полюсами ЛИС-системы с передаточной функцией $\frac{1}{H(z^{-1})}$ будут значения $\frac{1}{p_k}$, а значит $\max_k \left|\frac{1}{p_k}\right| > 1$. С учётом особенности реализации `scipy.signal.deconvolve` через деление многочленов, процедура развёртки не будет устойчивой.

Обойти неустойчивость можно, если решать задачу в два шага. На первом шаге вычисляется развёртка для $f[n] * h[n] * h[-n]$ и $h[n]$. Получим вспомогательный сигнал $p[n]$, который можно трактовать как оценку $f[n] * h[-n]$. На втором шаге необходимо вычислить развёртку $p[n] \approx f[n] * h[-n]$ и $h[-n]$, но это процедура при применении «в лоб» неустойчива. Однако в выражении $p[n] \approx f[n] * h[-n]$ можно из-

менить направление всех сигналов: $p[-n] \approx f[-n] * h[n]$ и вычислить устойчивую развёртку сигналов $p[-n]$ и $h[n]$. При этом получится обращённый по времени результат $f[-n]$, который необходимо перевернуть обратно.

Обратим внимание, что обращение во времени возможно, только если сигнал присутствует в памяти целиком. Применить данный прием для построения фильтра с обработкой в реальном времени нельзя.

Пример демонстрации неустойчивой развёртки вида $f[n] * h[n] * h[-n]$ приведён в файле `deconvolve.py`, который можно найти в репозитории курса.

2 Задание на лабораторную работу

2.1 Оценка искажающей импульсной характеристики

Скачайте из репозитория файл `lab7.py`, в котором будет выполняться лабораторная работа. Скачайте отсюда файл с данными `XX.npy`, где `XX` – номер вашего варианта.

Файл с данными представлен в виде матрицы, первая строка которой соответствует реализации $y[n]$, вторая строка — реализации $v[n]$, все оставшиеся строки — зашумленные реализации $h[n]$.

$y[n]$ — принятый (наблюдаемый) сигнал, в котором с помощью кода Морзе (см. Приложение 1) закодировано сообщение. Наблюдаемый сигнал является искажённым, поэтому непосредственное декодирование сообщения невозможно — сначала необходимо восстановить сигнал.

Для восстановления сигнала необходима дополнительная информация об искажениях, вносимых каналом связи. Для этого предоставлены сигналы $v[n]$ и $h[n]$.

$v[n]$ — одна из реализаций принимаемого сигнала в условиях отсутствия передаваемого (полезного) сигнала $x[n] = 0$, то есть содержит одну из возможных реализаций шума.

$h[n]$ — тысяча реализаций принимаемого сигнала при передаче единичного импульса $x[n] = \delta[n]$. Каждая строка соответствует очередной реализации.

Шаг 1. В основной функции `main` загрузите все имеющиеся сигналы:

```
data = np.load("?.npy")
y = np.ravel(data[0, :])
v = np.ravel(data[1, :])
h = data[2:, :]
```

Получите оценку искажающей импульсной характеристики $h[n]$. В случае белого шума усреднение по множеству реализаций может дать достаточно хорошую оценку импульсной характеристики искажающей системы. Результат «прихорошите»:

```
h[np.abs(h) < ?] = 0
h = np.trim_zeros(h, "b")
```

Запишите полученную оценку ИХ на бумаге в виде функции. Проанализируйте обратный фильтр с передаточной функцией вида $\frac{1}{H(z)}$ на предмет устойчивости.

2.2 Восстановление с помощью фильтра нижних частот и развёртки

Напишите реализацию функции `lowpass_reconstruct`. Функция должна принимать наблюдаемый сигнал $y[n]$ и импульсную характеристику искажающей системы $h[n]$ (в этом варианте восстановления параметры шума не нужны; лишь предполагается, что он белый). Результатом работы функции должен являться восстановленный сигнал.

Пусть одна точка в коде Морзе занимает M отсчётов. Верхняя частота в полезном сигнале определяется сигналом, состоящим из одних только точек, то есть гармоникой с периодом $2M$ отсчётов. Обозначим относительную частоту, соответствующую такой гармонике, как ω_0. Тогда основная энергия полезного сигнала сосредоточена в интервале частот $[0; \omega_0]$. Если шум белый, то в спектре он занимает всю полосу частот $[0; \pi]$. Тогда для реконструирования сигнала можно построить низкочастотный фильтр с частотой среза ω_0:

$$H_{\text{B}}(\omega) = \begin{cases} 1, & 0 < \omega < \omega_0; \\ 0, & \omega_0 < \omega < \pi. \end{cases}$$

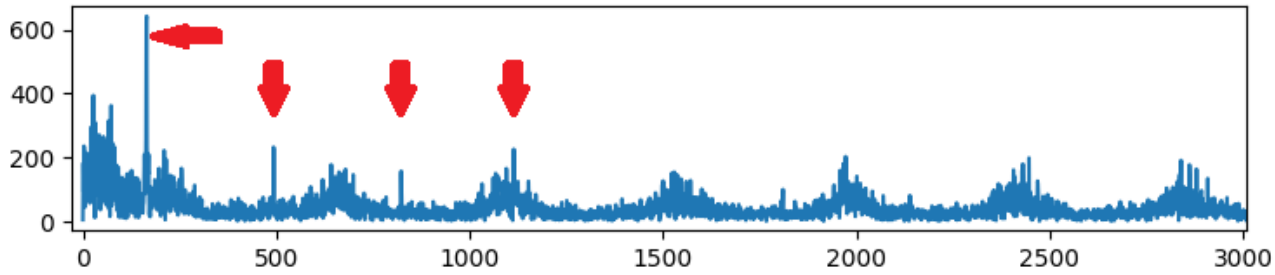
Такой фильтр уберёт высокочастотные составляющие, в которых в основном сосредоточен только шум, не трогая при этом низкочастотные составляющие, в которых сосредоточен в основном полезный сигнал. На область отрицательных частот фильтр дополняется симметричным образом в условиях требования вещественности $h_{\text{B}}[n]$.

Шаг 2.1. Проведите развертку наблюдаемого сигнала $y[n]$ с импульсной характеристикой $h[n]$ и получите оценку пока ещё зашумленного полезного сигнала $x_1[n]$.

Шаг 2.2. Постройте амплитудный спектр $|X_1[m]|$. При построении амплитудного спектра, чтобы избежать неинтересующего нас пика в области нулевой частоты (пропорциональному среднему значению сигнала) предварительно можно вычесть сред-

нее значение из всего сигнала. По позиции максимума в спектре определите размер одной точки Морзе M и соответствующую частоту среза ω_0.

Пример получающегося амплитудного спектра приведён на рисунке ниже. Обратите внимание, что помимо ярко выраженного пика с частотой ω_0, есть также пики на частотах $3\omega_0$, $5\omega_0$, $7\omega_0$ и т. д. Таким образом, для более надёжного определения значения ω_0 можно (но необязательно) воспользоваться кепстром. Однако в случае использования кепстра будьте готовы объяснить, почему не наблюдаются пики на частотах $2\omega_0$, $4\omega_0$ и т. д.

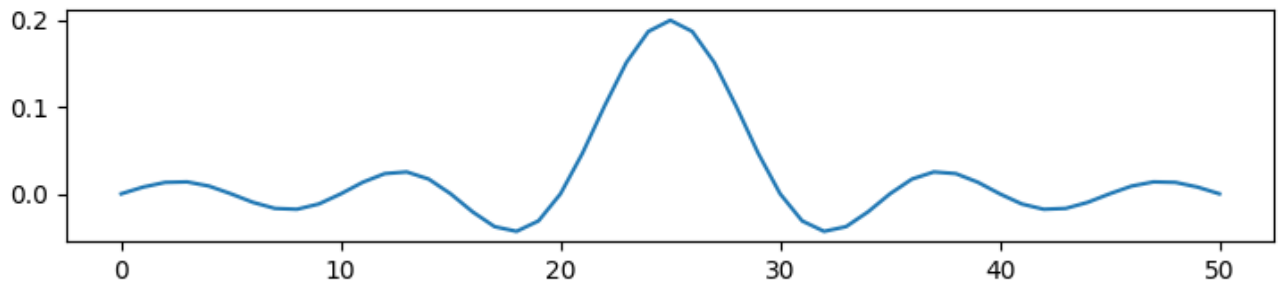


Шаг 2.3. С помощью обратного ДВПФ рассчитайте на бумаге вид импульсной характеристики $h_{\text{B}}[n]$ для идеального низкочастотного фильтра $H_{\text{B}}(\omega)$ с частотой среза ω_0. Получится физически нереализуемая БИХ-система, при этом $\lim\limits_{n\to\pm\infty} h_{\text{B}}[n] = 0$. Можно построить неидеальную КИХ-аппроксимацию такого фильтра, если искусственно ограничить область определения, положив $h_{\text{B}}[n] = 0$ при $n \notin [-(N \operatorname{div} 2); N \operatorname{div} 2]$, где N — нечётное число, определяющее размер фильтра (количество ненулевых отсчётов импульсной характеристики).

Чтобы фильтр сделать окончательно реализуемым, импульсную характеристику необходимо задержать (сдвинуть вправо) на $N \operatorname{div} 2$ отсчёта.

Для выполнения этого шага реализуйте вспомогательную функцию `build_lowpass` для расчёта низкочастотного КИХ-фильтра с частотой среза ω_0 и размером в N отсчётов.

Если всё сделано верно, то рассчитанная импульсная характеристика, например, для $\omega_0 = 0.2\pi$ и $n = 51$, должна выглядеть следующим образом:



Обратите внимание, что за счёт задержки импульсной характеристики для обеспечения физической реализуемости фильтра, применение такого фильтра будет вносить задержку на $N \operatorname{div} 2$ отсчёта.

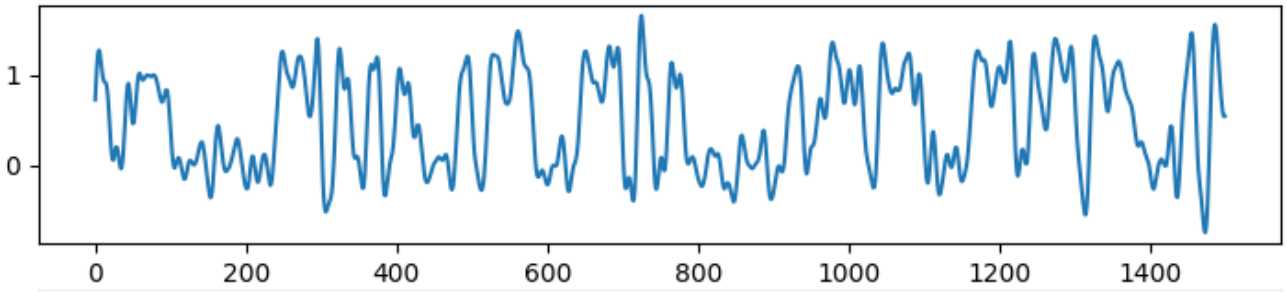
Шаг 2.4. Используя значение частоты среза ω_0 (из шага 2.2) и функцию `build_lowpass` (из шага 2.3), рассчитайте $h_{\mathrm{B}}[n]$. Размер фильтра положите равным 51.

Примените рассчитанный фильтр к сигналу $x_1[n]$ (из шага 2.1) — рассчитайте свёртку $x_1[n] * h_{\mathrm{B}}[n]$ с помощью `scipy.signal.convolve` и получите оценку полезного сигнала $\hat{x}[n]$ — основной результат работы функции `lowpass_reconstruct`.

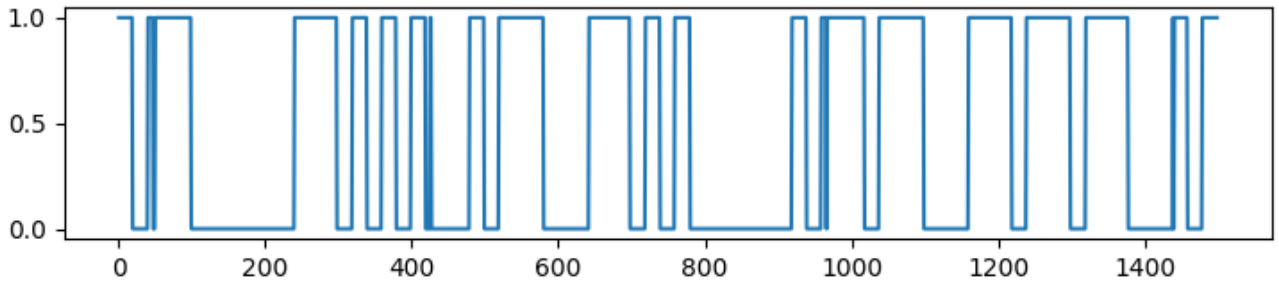
2.3 Декодирование кода Морзе и вычисление ошибки восстановления

Данный подраздел выполняется в функции `main`. Вызовите функцию `lowpass_reconstruct` и визуализируйте восстановленный сигнал.

Шаг 3.1. По сигналу $\hat{x}[n]$ восстановите закодированное с помощью кода Морзе сообщение. Если всё сделано верно, но сигнал $\hat{x}[n]$ должен выглядеть следующим образом:



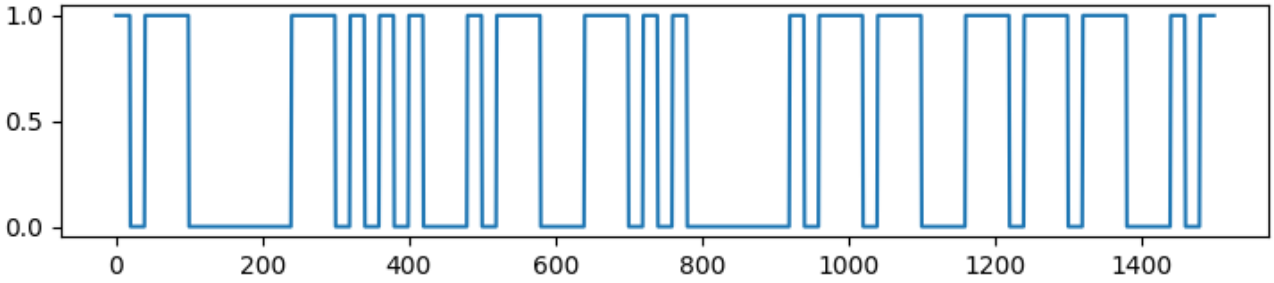
Можно немного упростить процесс декодирования, если провести пороговую обработку сигнала (все значения меньше 0,5 заменить на 0, все значения больше или равные 0,5 заменить на 1):



Обратите внимание, что восстановленный сигнал не является идеальным — например, в первом тире присутствует провал, а после трёх точек — ложный пик.

Далее восстановите закодированое сообщение. Разрешается сделать это как вручную (просто — визуально по графику), так и автоматически (сложнее — придётся подумать и написать какой-то код).

Шаг 3.2. Имея декодированное сообщение и значение M, постройте идеальный сигнал $x[n]$ с помощью функции `morse_encode` (реализация данной функции присутствует в данном вам шаблоне). Если вы использовали автоматическое восстановление с дополнительной фильтрацией на прошлом шаге, то сигнал с прошлого шага должен полностью совпадать с идеальным. Пример идеального сигнала приведён на рисунке ниже.



Рассчитайте среднеквадратичную ошибку (MSE) между идеальным сигналом $x[n]$ и восстановленным сигналом $\hat{x}[n]$ (из шага 2.4, до применения пороговой обработки).

Учтите, что $\hat{x}[n]$ является сдвинутым по сравнению с $x[n]$ из-за задержки, вносимой восстанавливающим фильтром. То есть при сравнении сигналов их предварительно нужно выровнять по времени (компенсировать задержку). Кроме того, наблюдаемый сигнал $y[n]$ (а значит, и восстановленный сигнал) дан с некоторым запасом по длине. При сравнении обрежьте восстановленный сигнал до размера идеального сигнала.

2.4 Квазиоптимальное восстановление с помощью фильтра Винера

Рассчитайте и примените квазиоптимальный фильтр, построенный на той же области, что и низкочастотный фильтр из подраздела 2.2: $D = \{-(N \text{ div } 2), ..., N \text{ div } 2\}, N = 51$.

Обратите внимание на два момента. Во-первых, при построении фильтра на такой области будут нужны отсчёты АКФ в области нуля. Для того, чтобы избежать различных краевых эффектов при развёртке, формируйте оценки АКФ так, чтобы нулевой отсчёт АКФ оказывался примерно в центре сигнала (т. е. задерживайте АКФ как минимум на величину порядка $N \text{ div } 2$). Во-вторых, в условиях построения фильтра Винера необходимо получить ненормированные оценки АКФ — такие, что $R[0] = \sigma^2$.

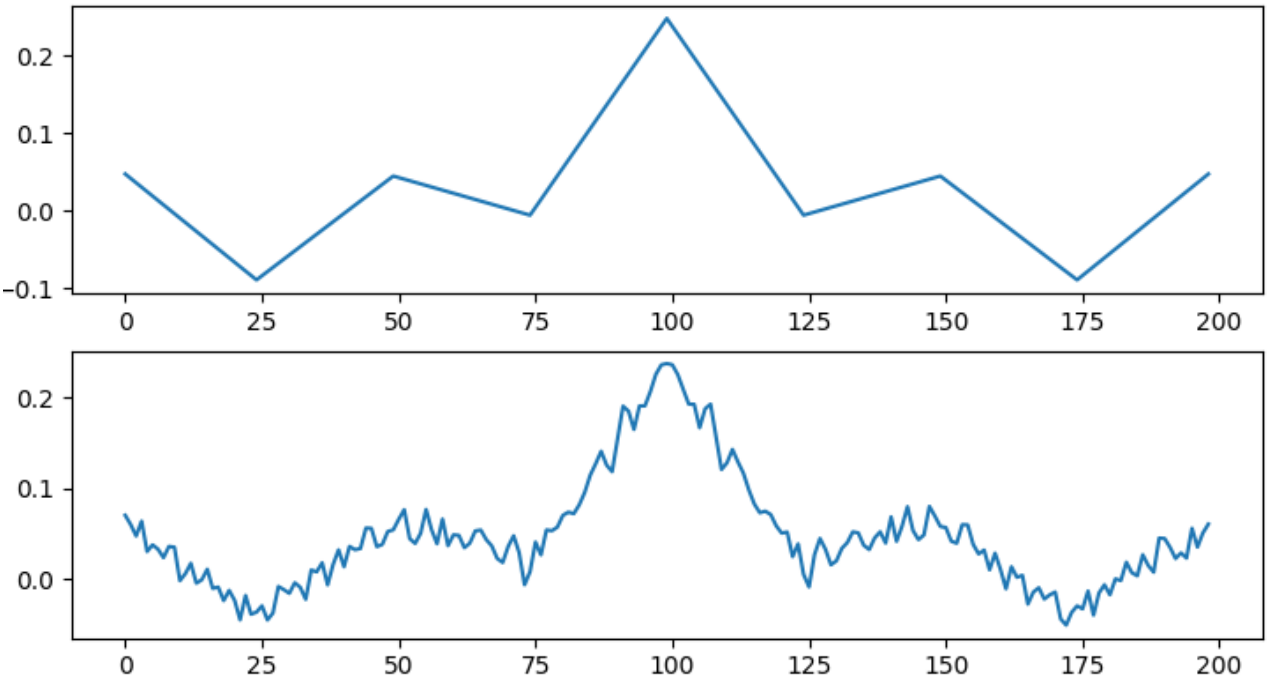
Шаги данного подраздела выполняются в функции `suboptimal_reconstruct`.

Шаг 4.1. С помощью `statsmodels.tsa.stattools.acf` и `numpy.var` оцените $R_y[n]$ и $R_v[n]$ по имеющимся реализациям.

Шаг 4.2. Используя известные выражения для $R_{xy}[n]$ и $R_x[n]$ в условиях независимости полезного сигнала и шума, оцените эти АКФ с помощью развёртки.

Так как на шаге 3.2 был получен идеальный полезный сигнал, можно (но необязательно) построить оценку АКФ по самому полезному сигналу и сравнить её с оценкой, полученной на данном шаге. Отметим, что такое сравнение делается исключительно для факультативной проверки правильности выполнения данного шага и не требуется для решения поставленной задачи (ведь строится фильтр для получения оценки полезного сигнала, самого полезного сигнала пока нет).

Если всё сделано правильно, то оценки АКФ (по полезному сигналу и через развёртку по АКФ наблюдаемого сигнала) должны выглядеть примерно так:



Обратите внимание, что оценки задержаны примерно на 100 отсчётов и построены на области D с запасом по краям.

Шаг 4.3. Составьте систему линейных уравнений Винера-Хопфа, решите систему с помощью `numpy.linalg.solve` и примените (вычислите свёртку) построенный фильтр к наблюдаемому сигналу для получения оценки $\hat{x}[n]$. Обратите внимание, что здесь аналогичным образом нужно компенсировать задержку, вызванную обеспечением физической реализуемости восстанавливающего фильтра.

Шаг 4.4. По оценке дисперсии полезного сигнала $R_x[0]$ и вычисленной $h_{\text{В}}[n]$ оцените погрешность восстановления по формуле из подраздела 1.1.

2.5 Вычисление ошибки восстановления

Шаг 5. В функции `main` вызовите функцию `suboptimal_reconstruct`. Рассчитайте среднеквадратичную ошибку. Как и в прошлом случае, ошибку нужно считать между идеальным полезным сигналам (шаг 3.2) и оценкой, непосредственно получаемой сразу после применения квазиоптимального фильтра (но до всяких пороговых обработок).

3 Формат сдачи

Помимо файла (файлов) с выполненной лабораторной работой, предоставить в письменном виде на бумаге:

1) аналитический вид искажающей $h[n]$ и доказательство устойчивости обратного фильтра $\dfrac{1}{H(z)}$ (шаг 1);

2) значение $M[\text{отсчёты}]$ – размер одной точки кода Морзе и соответствующее значение частоты среза ω_0 $\left[\dfrac{\text{рад}}{\text{отсчёт}}\right]$ (шаг 2.2);

3) аналитический вид $h_{\text{В}}[n]$ идеального фильтра нижних частот с частотой среза ω_0 (шаг 2.3);

4) текст восстановленного сообщения (шаг 3.1);

5) значение среднеквадратичной ошибки $\sigma^2_{e_1}$ при восстановлении сигнала с помощью фильтра нижних частот (шаг 3.2, рассчитывается с использованием идеального и восстановленного сигналов);

6) значение среднеквадратичной ошибки $\sigma^2_{e_2}$ при восстановлении сигнала с помощью фильтра Винера (шаг 5, рассчитывается с использованием идеального и восстановленного сигналов);

7) оценка MSE-ошибки σ^2_e при восстановлении сигнала с помощью фильтра Винера (шаг 4.4, рассчитывается по соответствующему уравнению без использования непосредственных реализаций сигналов, лишь с использованием их статистических характеристик и $h_{\text{В}}[n]$).

Если всё сделано верно, то $\sigma^2_{e_1} > \sigma^2_{e_2}$. Значения $\sigma^2_{e_2}$ и σ^2_e в общем случае могут отличаться друг от друга в обе стороны (и достаточно сильно, это нормально). Примерный диапазон верного значения $\sigma^2_{e_1}$ — $[0.04 \ldots 0.07]$.

Приложение 1. Код Морзе

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.