

Лекция 1: Цифровая фильтрация (продолжение) и типы в языке программирование Си

Д. А. Караваев

Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М. А. Бонч-Бруевича

Факультет РТС, Кафедра РОС

Факультатив «Программирование в ЦОС»

Осень 2019

14.10.2019 Санкт-Петербург

В радиотехнических приложениях ЦОС повсеместно используется комплексное IQ представление сигнала:

$$U[n] = I[n] + jQ[n] = a[n] \cos(\varphi[n]) + a[n]j \sin(\varphi[n]), \quad (1)$$

где $a[n]$, $\varphi[n]$ - изменение амплитуды и фазы сигнала, а $U[n]$ - *комплексная огибающая* (дискретизированная) несущего колебания.

Будем рассматривать произвольные комплексные сигналы (необязательно имеющие IQ интерпретацию):

$$s[n] = \operatorname{Re}\{s[n]\} + j\operatorname{Im}\{s[n]\}, \quad s[n] \in \mathbb{C}. \quad (2)$$

Одним из первичных этапов цифровой обработки радиосигналов является перенос *вещественного* сигнала с АЦП с **промежуточной частоты** (ПЧ) на 0 (получение *комплексной огибающей*) и установление необходимой F_s путём децимации:

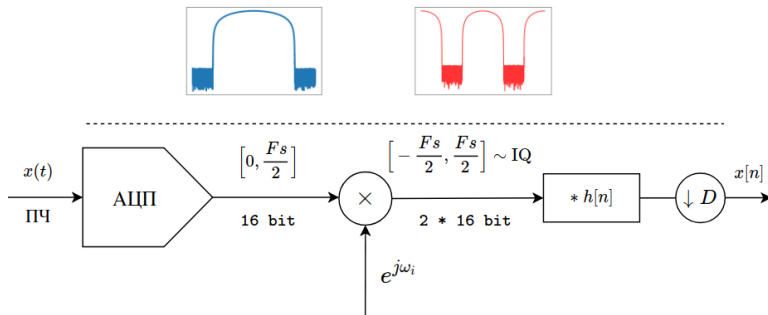


Рис.: Схема Digital Down Converter

КИХ-фильтр для комплексных сигналов

Пусть входной сигнал КИХ-фильтра $x[n]$ является комплексным, а ИХ по-прежнему вещественная $h[n]$ длины T , тогда:

$$\begin{aligned} y[n] &= x[n] * h[n] = \sum_{k=0}^{T-1} h[k]x[n-k] = \\ &= \sum_{k=0}^{T-1} h[k]\operatorname{Re}\{x[n-k]\} + j \sum_{k=0}^{T-1} h[k]\operatorname{Im}\{x[n-k]\} = \\ &= \operatorname{Re}\{x[n]\} * h[n] + j\operatorname{Im}\{x[n]\} * h[n], \end{aligned} \tag{3}$$

Вывод: Комплексная фильтрация \implies фильтрация вещественной и мнимой части по отдельности.

Псевдокод алгоритма комплексной свёртки

Вход : $\hat{x}[n] \in \mathbb{C}$ - вх. сигнал длины $N + T - 1$, $h[n] \in \mathbb{R}$ - ИХ длины T .

Выход: $y[n] \in \mathbb{C}$ - вых. сигнал длины N .

for $n \leftarrow 0$ **to** $N - 1$ **do**

$\overline{\text{Re}\{y[n]\}} \leftarrow 0$

$\text{Im}\{y[n]\} \leftarrow 0$

$m \leftarrow n + (T - 1)$

for $k \leftarrow 0$ **to** $T - 1$ **do**

$\overline{\text{Re}\{y[n]\}} \leftarrow \overline{\text{Re}\{y[n]\}} + \text{Re}\{\hat{x}[m - k]\}h[k]$

$\text{Im}\{y[n]\} \leftarrow \text{Im}\{y[n]\} + \text{Im}\{\hat{x}[m - k]\}h[k]$

end

end

Замечание: Необходимо разобраться, как в языке программирования Си можно представить комплексные числа.

Фильтрация без усиления

В приложениях часто требуется, чтобы амплитуда в полосе пропускания фильтра вых. сигнала $y[n]$ *значительно* не изменялась по сравнению с вх. сигналом $x[n]$:

$$h[n] \leftarrow \frac{h[n]}{|H(e^{j\omega_0})|}, \quad \omega_0 = 0, \implies \left| \sum_{k=0}^{T-1} h[k] \right| = 1,$$
$$H(e^{j\omega}) = \sum_{k=0}^{T-1} h[k] e^{j\omega k}, \quad (4)$$
$$|H(e^{j\omega_0})| = \left| \sum_{k=0}^{T-1} h[k] e^{j\omega_0 k} \right| = \left| \sum_{k=0}^{T-1} h[k] \right|$$

Вопрос: Что делать с условием (4) для целых чисел (типа `int`)?

Приведение ИХ

Решение:

$$h_{\text{int}}[n] = \lfloor \text{int}_{\text{max}} \times h_{\text{float}}[n] \rfloor \implies y_{\text{int}}[n] = \frac{1}{\text{int}_{\text{max}}} (x_{\text{int}}[n] * h_{\text{int}}[n]).$$

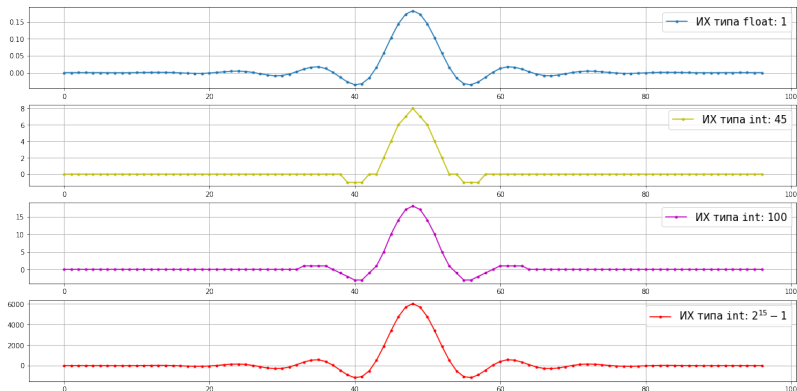


Рис.: Динамика значений ИХ

Представление комплексных чисел в языке Си

```
/* В языке Си есть встроенные конструкции для комплексных чисел,  
 * но мы их использовать не будем:) */  
float z[2]; /* Массив с вещественной и мнимой частью. */  
z[0] = 1.0; /* Вещественная часть == 1.0. */  
z[1] = 2.0; /* Мнимая часть == 2.0. */  
/* Комплексный массив: */  
float s[10][2]; /* Двумерный массив: 1 - время, 2 - в. и м. части! */  
s[4][0] = 12.48; /* Вещественная часть 4-ого отсчёта. */  
s[4][1] = 1.618; /* Мнимая часть 4-ого отсчёта. */  
/* Иногда такие массивы "линеаризуются": */  
float sl[20]; /* Одномерный массив. */  
sl[2 * 4 + 0] = 12.48; /* Вещественная часть 4-ого отсчёта. */  
sl[2 * 4 + 1] = -1.618; /* Мнимая часть 4-ого отсчёта. */  
/* "Целые" комплексные числа: */  
int zi[2]; /* Аналогично и целые комплексные массивы. */
```


Перечисления в языке Си

```
/* В языке Си можно задавать особый тип - перечисление  
 * (набор уникальных именованных целых констант): */
```

```
enum color_t
```

```
{  
    RED = 0, GREEN = 1, BLUE = 2,  
};
```

```
/* Мы можем сделать так: */
```

```
enum DSP_part_t
```

```
{  
    RE = 0, IM = 1  
};
```

```
/* для обращения к в. и м. частям комп. числа: */
```

```
int n = 3;
```

```
s[n][RE] = 1.4;
```

```
s[n][IM] = 1.8;
```

Числовые типы в языке Си

```
/* Существует разновидность типов целых чисел: */  
char s_byte; /* [-2^7, 2^7 - 1]. */  
short s_word; /* [-2^15, 2^15 - 1]. */  
int s_dword; /* [-2^31, 2^31 - 1]. */  
long s_qword; /* [-2^63, 2^63 - 1]. */  
  
/* Для получения беззнаковых версий добавляем unsigned: */  
unsigned char u_byte; /* [0, 2^8 - 1]. */  
unsigned short u_word; /* [0, 2^16 - 1]. */  
unsigned int u_dword; /* [0, 2^32 - 1]. */  
unsigned long u_qword; /* [0, 2^64 - 1]. */  
  
/* Так же есть два типа для чисел с дробной частью: */  
float s; /* Одинарная точность. */  
double d; /* Двойная точность. */
```

Директивы typedef и sizeof

```
/* Создание псевдонима типа: */
typedef char          int8_t;   /* [-2^7, 2^7 - 1]. */
typedef short         int16_t;  /* [-2^15, 2^15 - 1]. */
typedef int           int32_t;  /* [-2^31, 2^31 - 1]. */
typedef long          int64_t;  /* [-2^63, 2^63 - 1]. */
typedef unsigned char uint8_t;  /* [0, 2^8 - 1]. */
typedef unsigned short uint16_t; /* [0, 2^16 - 1]. */
typedef unsigned int  uint32_t; /* [0, 2^32 - 1]. */
typedef unsigned long uint64_t; /* [0, 2^64 - 1]. */
typedef unsigned long size_t;   /* Tun size_t. */

/* Создание переменной: */
int32_t x = 0;

/* Сколько байт в типе/переменной: */
int bytes = sizeof(uint32_t); /* == 4.*/
int truth = (bytes == sizeof(x));
```

Приведение типов (0-вое приближение)

```
/* Зачастую необходимо преобразовать значение из  
* одного типа в другой */  
char byte = 10;  
/* Для избегания переполнения: */  
int16_t sample = (int16_t)byte * 100; /* == 1000. */  
/* На самом деле в Си это происходит по-умолчанию: */  
int16_t another = byte * byte;  
/* или */  
double x = sample;  
/* ?: */  
float alpha = (float)101 / byte;
```

Вывод: $\left| \sum_{k=0}^{T-1} h_{\text{int}}[k] \right|$ можно нормировать к \max_{int16} , а результат $h_{\text{int16}}[k] \times x_{\text{int16}}[n - k] \leftarrow y_{\text{int32}} \implies y_{\text{int16}} = y_{\text{int32}} / \max_{\text{int16}}$.

Прототип функции: source/convolve.c

```
/*!
 * |param[in] xh - входной сигнал, дополненный  $T - 1$  нулями.
 * |param[out] y - выходной сигнал.
 * |param[in] N - длина исходного входной сигнала.
 * |param[in] h - ИХ.
 * |param[in] T - длина ИХ.
 * |return 0 - успех, -1 - нереализована.
 */
int DSP_convolve_complex_int(const int16_t* xh, int16_t* y, size_t N,
                             const int16_t* h, size_t T)
{
    /* Ваш код. */
    return 0;
}
```

Результат комплексной свёртки

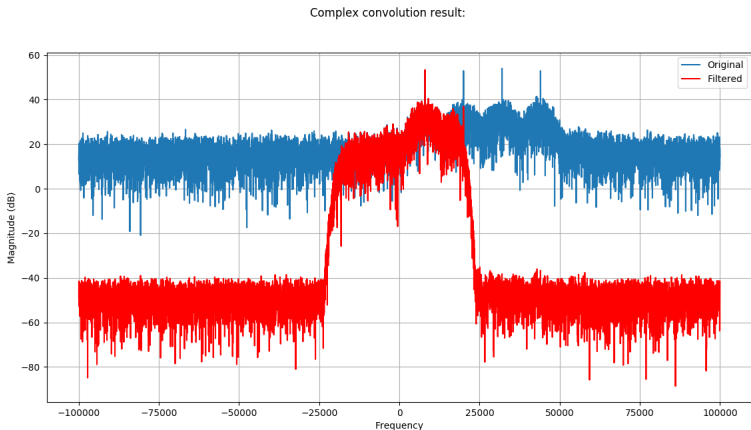


Рис.: Результат комплексной целочисленной фильтрации 4FSK сигнала.
Команда для отображения: `python3 ../scripts/plot_complex_int.py`

Спасибо за внимание!