Моделирование цифровой обработки сигналов в MATLAB

Часть 6. Моделирование структур цифровых фильтров с фиксированной точкой программными средствами MATLAB: квантование воздействия и вычисление реакции

Алла СОЛОНИНА

Ранее [9] были перечислены источники ошибок квантования в цифровых фильтрах (ЦФ) с фиксированной точкой (ФТ) и составлена нелинейная модель ЦФ с ФТ, предназначенная для компьютерного моделирования. В [9], [10] была приведена методика составления и свойства моделей КИХ- и БИХ-фильтров с ФТ, на основе которых анализируются их основные характеристики.

Следующий этап в исследовании эффектов квантования в ЦФ с ФТ на основе нелинейной модели [9] программными средствами MATLAB связан с созданием модели квантованного воздействия, вычислением реакции и сравнением ее с реакцией в отсутствии квантования на основе объективных критериев. Этим вопросам посвящена настоящая статья.

Моделирование квантования в АЦП

Квантование воздействия в АЦП (далее — квантование) в MATLAB может моделироваться одним из следующих способов:

- на основе объекта quantizer с помощью функции quantize;
- на основе объекта fi.

Моделирование квантования на основе объекта quantizer

При моделировании квантования на основе объекта quantizer (Quantizer Object) сначала создается данный объект одним из следующих основных способов:

```
q=quantizer('name1',value1,'name2',value2,...)
```

или

```
q=quantizer(struct)
```

Здесь q — имя объекта quantizer; 'name1','name2',... — свойства объекта quantizer; value1,value2,... — значения свойств объекта quantizer; struct — имя массива записей (struct array), поля которого отображают свойства объекта quantizer.

Выходным параметром является объект q (массив записей — struct array) со списком свойств (полей). Их полный список выводится с помощью функции get(q), а основной — по имени q.

Доступные пользователю свойства устанавливаются с помощью функции:

```
set(q,'name1',value1,'name2',value2,...)
```

В таблице 1 представлены основные доступные пользователю свойства объекта quantizer, а в таблице 2 — четыре важнейших из множества свойств, доступных только для чтения.

Таблица 1. Доступные пользователю свойства объекта quantizer

Свойство	Назначение
Mode	Отображает тип квантованных данных с интересующими нас значениями: • 'double' — данные с ПТ типа double; • 'fixed' (по умолчанию) — данные с ФТ; • 'ufixed' — беззнаковые данные с ФТ
RoundMode	Отображает метод округления при квантовании данных (представлении в заданном формате) и может принимать значения: • 'round' — округление до ближайшего (округление с избытком): числу присваивается значение ближайшего уровня квантования, и если оно попадает точно на границу между соседними уровнями — ближайшего верхнего для положительного числа и для модуля отрицательного; • 'convergent' — округление до ближайшего четного: числу присваивается значение ближайшего уровня квантования, и если оно попадает точно на границу между соседними уровнями — ближайшего, соответствующего четному числу; • 'fix' — округление в направлении нуля, тождественно усечению; • 'ceil' — округление в направлении ∞: отрицательное число усекается, а положительному присваивается значение ближайшего верхнего уровня; • 'floor' (по умолчанию) — округление в направлении −∞: положительное число усекается, а модулю отрицательного присваивается значение ближайшего верхнего уровня
OverflowMode	Управляет переполнением при сохранении данных в формате слова и может принимать значения: • 'saturate' (по умолчанию) — при переполнении реализуется арифметика насыщения: результат автоматически заменяется максимально возможным по модулю для формата словом при ФТ; • 'wrap' — при переполнении реализуется модульная арифметика: результат автоматически заменяется своим значением по модулю максимального целого числа
Format	Отображает формат данных для выбранного значения свойства Mode. Формат данных типа 'fixed' и 'ufixed' отображается двухэлементным вектором, по умолчанию [16 15], в котором первый элемент соответствует длине слова, а второй — длине его дробной части

Пример 1

Создать двумя способами объект quantizer с именем q и заданными свойствами и сохранить его на диске для дальнейшего использования.

1. Первый способ:

```
>> q=quantizer('Mode','fixed','RoundMode','convergent',...
'OverflowMode','saturate','Format',[16 15])
>> q =
            DataMode  = fixed
           RoundMode  = convergent
        OverflowMode  = saturate
              Format  = [16 15]
```

Таблица 2. Свойства объекта quantizer только для чтения

Свойство	Назначение
max	Максимальное значение данных до квантования
min	Минимальное значение данных до квантования
noverflows	Количество переполнений
nunderflows	Количество потерь значимости (антипереполнений), возникающих, когда малое значение данных не может быть представлено в заданном формате и потому заменяется нулем

Таблица 3. Функции MATLAB для объектов quantizer

Функция	Назначение
qn=copyobj(q)	Создает новый независимый объект qn, сохраняющий все свойства объекта q, при этом изменение свойств объекта qn никак не влияет на свойства q. При использовании оператора присваивания qn=q изменение свойств объекта qn приводит к автоматическому изменению свойств объекта q
xnum= bin2num(q,xbin)	Преобразует числа xbin в числа xnum (xnum — квантованные согласно свойствам объекта q десятичные числа); xbin — указанные в апострофах двоичные числа, представляющие собой двоичный эквивалент* чисел xnum): >> load q >> xbin=['0111110001100110';'1000110001100110']; >> xnum=bin2num(q,xbin) xnum = 0.9719 −0.9031
xnum= hex2num(q,xhex)	Преобразует числа xhex в числа xnum (xhex — указанные в апострофах шестнадцатеричные числа, представляющие собой шестнадцатеричный эквивалент** чисел xnum): >> load q >> xhex=['7C66';'8C66']; >> xnum=hex2num(q,xhex) xnum = 0.9719 −0.9031 Числа 7C66 и 8C66 представляют собой соответственно краткую запись шестнадцатеричными цифрами двоичных эквивалентов 0111110001100110 и 1000110001100110 (см. функцию bin2num)
xbin= num2bin(q,xnum)	Преобразует числа xnum в числа xbin: >> xnum=[0.9719;−0.9031]; >> xbin=num2bin(q,xnum) xbin = 0111110001100111 1000110001100111
xhex= num2hex(q,xnum)	Преобразует числа xnum в числа xhex: >> load q >> xnum=[0.9719;−0.9031]; >> xhex=num2hex(q,xnum) xhex = 7c67 8c67
xint= num2int(q,xnum)	Преобразует числа xnum в числа xint (xint — десятичные числа, представляющие собой целочисленный эквивалент*** чисел xnum): >> load q >> xnum=[0.9719;−0.9031]; >> xint=num2int(q,xnum) xint = 31847 −29593 Десятичные числа 31847 и −29593 соответствуют целым двоичным числам со знаком 0111110001100110 и 1000110001100110 (см. функцию bin2num)

Примечания. * — Двоичным эквивалентом квантованного числа называют его побитовую запись в слове при заданном формате, включая знаковый бит.
** — Шестнадцатеричным эквивалентом квантованного числа называют краткую запись его двоичного эквивалента, когда каждые его четыре бита, включая знаковый, представляются шестнадцатеричной цифрой.
*** — Целочисленным эквивалентом квантованного числа называют запись его двоичного эквивалента в виде целого десятичного числа со знаком, получаемую по правилам перевода целого двоичного числа в десятичное.

2. Второй способ (свойства объекта q точно такие же):

```
>> struct.Mode='fixed';
>> struct.RoundMode='convergent';
>> struct.OverflowMode='saturate';
>> struct.Format=[16 15];
>> q=quantizer(struct);
>> save q
```

Объекты quantizer могут использоваться в качестве входных параметров многих функций MATLAB, основные из которых, связанные с копированием квантователя и преобразованием чисел, приведены в таблице 3 (использован сохраненный в примере 1 квантователь q).

На основе созданного объекта quantizer выполняется непосредственно квантование с помощью функции:

```
xq=quantise(q,x)
```

Здесь q — имя объекта quantizer; x — исходные неквантованные данные: вектор или матрица; xq — квантованные данные; размеры x и xq совпадают.

В результате квантования обновляются свойства объекта q, доступные только для чтения, которые рекомендуется проверять.

Для иллюстрации квантования удобно, как это предлагается в MATLAB, квантовать элементы вектора, изменяющиеся линейно в заданном диапазоне, которые можно генерировать с помощью функции:

```
x=linspace(xb,xf,N)
```

Здесь x — вектор с линейно меняющимися элементами; xb, xf — значения первого и последнего элементов вектора x; N — число элементов вектора (по умолчанию 100).

Пример 2

На основе объекта quantizer с именем q и заданными свойствами (пример 1) выполнить квантование элементов вектора x при xb = −2 и xf = 2. Повторить процедуру для объекта qq (копии объекта q) со свойством OverflowMode = wrap (модульной арифметикой). Соответствующим векторам с квантованными элементами присвоить имена xq1 и xq2. Построить графики зависимостей x от xq1 и x от xq2, отображающие характеристики квантователей при разных типах арифметики и иллюстрирующие суть различий между арифметикой насыщения и модульной арифметикой (рисунок):

```
>> load q
>> xb=−2; xf=2; x=linspace(xb,xf); xq1=quantize(q,x);
Warning: 50 overflows.
> In embedded.quantizer.quantize at 68
>> subplot(2,1,1), plot(x,xq1), grid,xlabel('x'), ylabel('xq1'),...
title('OverflowMode=saturate')
>> qq=copyobj(q); set(qq,'OverflowMode','wrap')
>> xq2=quantize(qq,x);
>> subplot(2,1,2), plot(x,xq2), grid,xlabel('x'), ylabel('xq2'),...
title('OverflowMode=wrap')
```

Моделирование квантования на основе объекта fi

При моделировании квантования на основе объекта fi (Fixed-point Numeric Object) достаточно создать данный объект:

```
xin=fi(v,s,w,f)
```

Здесь xin — имя объекта fi; v, s, w, f — входные параметры объекта fi; v — исходные неквантованные данные: вектор или матрица; w — длина слова для данных с ФТ (по умолчанию 16); f — длина дробной части слова для данных с ФТ (по умолчанию 15); s — скаляр, принимающий значения 1 (true) или 0 (false); при s = 0 (по умолчанию) формируются числа с ФТ без знака (беззнаковые — положительные), а s = 1 — со знаком и появляется возможность управлять длиной f дробной части в слове длиной w.

Пример 3

Создать объект fi с именем xin и входными параметрами w = 16, f = 15, s = 1 и выполнить квантование элементов вектора x:

```
>> xb=-2; xf=2; x=linspace(xb,xf);
>> xin=fi(x,1,16,15);
```

Выходным параметром является объект xin — массив записей (struct array), поля которого отображают свойства объекта. Их удобно разделить на три группы:

- Data Properties (свойства данных) — это группа полей, отображающих различные виды представления данных с ФТ, основными из которых являются:

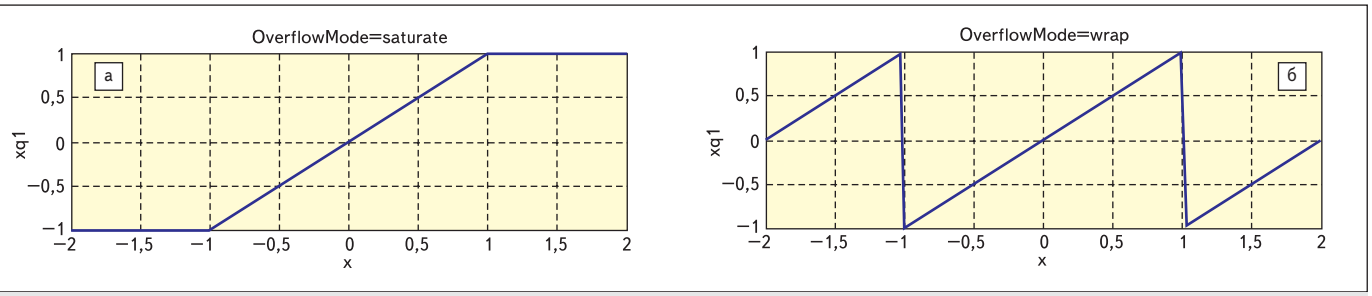

Рисунок. Характеристики квантователей при использовании: а) арифметики насыщения; б) модульной арифметики

  - xin.data — квантованные согласно входным параметрам объекта fi десятичные числа;
  - xin.double — квантованные согласно входным параметрам объекта fi десятичные числа типа double (при выводе не отличаются от xin.data);
  - xin.bin — двоичные эквиваленты чисел xin.data;
  - xin.hex — шестнадцатеричные эквиваленты чисел xin.data;
  - xin.int — целочисленные эквиваленты чисел xin.data.
- fimath Properties, где fimath — имя поля, в свою очередь, объединяющего группу полей (свойств) объекта xin, связанных с особенностями арифметики при представлении данных с ФТ, а также при работе с двумя объектами fi.
- numerictype Properties, где numerictype — имя поля, в свою очередь, объединяющего группу полей (свойств) объекта xin, связанных с типом данных, форматом, масштабированием и т. п., предназначенных только для чтения.

Пример 4

Создать объект fi с именем xtest и входными параметрами w = 16, f = 15, s = 1 и выполнить квантование элементов вектора a:

```
>> a=[0.9719 –0.9031];
>> xtest=fi(a,1,16,15);
```

Вывести квантованные значения, а также их двоичные, шестнадцатеричные и целочисленные эквиваленты, и сравнить результаты с полученными в таблице 3:

```
>> xtest.data
ans = 0.9719 –0.9031
>> xtest.bin
ans =
0111110001100111 1000110001100111
>> xtest.hex
ans = 7c67 8c67
>> xtest.int
ans = 31847 –29593
```

Пример 5

Для созданного в примере 3 объекта xin вывести список основных свойств (полей) группы fimath Properties:

```
>> xin.fimath
ans =
          RoundMode: round
       OverflowMode: saturate
        ProductMode: FullPrecision
 MaxProductWordLength: 128
            SumMode: FullPrecision
  MaxSumWordLength: 128
        CastBeforeSum: true
```

Для данных с ФТ в группе fimath Properties представляют интерес два поля (два свойства) объекта xin:

- xin.RoundMode — отображает метод округления при представлении данных с ФТ (свойство RoundMode в табл. 1);
- xin.OverflowMode — отображает режим переполнения при сохранении данных в формате слова (свойство OverflowMode в табл. 1).

Остальные поля отображают особенности арифметики при работе с двумя объектами fi.

Присвоим свойству RoundMode значение 'convergent' и сохраним объект xin на диске:

```
>> xin.RoundMode='convergent';
>> save xin
```

Пример 6

Для сохраненного на диске объекта xin (пример 5) вывести список основных свойств (полей) группы numerictype Properties:

```
>> load xin
>> xin.numerictype
ans =
   DataTypeMode: Fixed-point:  binary point scaling
                 Signed:  true
                 WordLength:  16
                 FractionLength:  15
```

Свойство DataTypeMode отображает режим масштабирования данных с ФТ. Его значение Fixed-point: binary point scaling соответствует отсутствию масштабирования данных с ФТ (о нем см. далее).

Остальные значения полей (свойств) объекта xin согласованы с входными параметрами объекта fi и предназначены только для чтения.

Полные списки свойств (полей) объекта xin (пример 5), относящихся к группам fimath Properties и numerictype Properties, выводятся с помощью функций get(xin.fimath) и get(xin.numerictype), а общий список свойств — с помощью функции get(xin). В целях экономии места эти списки не приводятся.

Среди дополнительных свойств в группе numerictype Properties представляют интерес дополнительные свойства, связанные с масштабированием данных с ФТ. Определяющим является свойство Scaling.

Пример 7

Для сохраненного на диске объекта xin (пример 5) вывести значение свойства Scaling в группе numerictype Properties:

```
>> Scaling=get(xin.numerictype,'Scaling')
Scaling = BinaryPoint
```

При значении свойства Scaling: 'BinaryPoint' (пример 6), когда масштабирование отсутствует, представляют интерес следующие дополнительные свойства:

- FixedExponent — отображает показатель степени при определении веса младшего бита $2^{\text{FixedExponent}}$ в заданном формате данных с ФТ;
- Slope — отображает точность представления дробной части данных с ФТ — вес младшего значащего бита $2^{\text{FixedExponent}}$ (шаг квантования).

Пример 8

Для сохраненного на диске объекта xin (пример 5) определить вес младшего бита и сравнить его с выведенным значением свойства Slope:

```
>> 2^(xin.FixedExponent)
ans = 3.0518e–005
>> Slope=get(xin.numerictype,'Slope')
Slope = 3.0518e–005
```

Свойства группы numerictype Properties невозможно изменить после создания объекта fi. Для того чтобы появилась возможность управлять ими при создании объекта, следует использовать другие форматы описания объекта fi.

Например, для масштабирования данных с ФТ, когда их значения можно будет изменять согласно формуле:

$$a_{\text{data}} = a_{\text{int}} \times \text{slope} + \text{bias}, \qquad (1)$$

где a_{data} — десятичное число с ФТ (xin.data); a_{int} — целочисленный эквивалент (xin.int) десятичного числа a_{data}; bias — смещение; slope — коэффициент масштабирования, равный:

$$\text{slope} = 2^{\text{FixedExponent}}, \qquad (2)$$

следует выбрать формат:

```
xin=fi(v,s,w,slope,bias)
```

Здесь slope соответствует свойству Slope и управляет, согласно (1), масштабированием данных, а именно: распределением битов в слове длиной WordLength (свойство объекта fi). При slope = $2^{\text{FixedExponent}}$ длина дробной части в WordLength автоматически становится равной FractionLength = –FixedExponent, а старшие биты, длина которых составляет (WordLength – FractionLength – 1), интерпретируются как целая часть двоичного числа; bias соответствует свойству Bias (выводится по команде get) и управляет, согласно (1), смещением bias.

При выборе данного формата описания объекта fi изменяются свойства создаваемого объекта xin, которые рекомендуется проверять. В частности, свойства DataTypeMode и Scaling принимают значения:

```
DataTypeMode:  'Fixed-point: slope and bias
      Scaling:  'SlopeBias'
```

Пример 9

Создать объект fi с именем x1 и параметрами w = 8, f = 7, s = 1 и выполнить квантование элементов вектора a:

```
>> a=[0.5555555 2.5555555];
>> x1=fi(a,1,8,7);
>> x1.data
ans = 0.5547        0.9922
>> x1.bin
ans = 01000111      01111111
>> x1.int
ans = 71 127
```

Как видим, число 2.5555555, большее единицы, в данном формате не может быть представлено, и для него использована арифметика насыщения: присвоено максимальное в данном формате значение 0,9922.

Рассмотрим, что изменится при значении slope = 2^(–5), когда на дробную часть FractionLength будет отведено не 7, а 5 битов, и, соответственно, на целую часть — 2 бита (значение bias остается равным 0):

```
>> a=[0.5555555 2.5555555];
>> slope=2^(–5);
>> bias=0;
>> x2=fi(a,1,8,slope,bias);
>> x2.data
ans = 0.5625        2.5625
>> x2.bin
ans = 00010010      01010010
>> x2.int
ans = 18            82
```

Новое значение параметра slope изменило и сами десятичные числа с ФТ, и их двоичный и целочисленный эквиваленты. В двоичных эквивалентах биты, интерпретируемые как целая часть двоичного числа, выделены полужирным шрифтом. Используя правила перевода десятичных чисел (целых и дробных) в двоичные, нетрудно убедиться в соответствии чисел x2.data и их двоичных эквивалентов. Однако целочисленные эквиваленты вычисляются на основании новых двоичных эквивалентов по «старому» правилу, в чем также легко убедиться:

```
>> 0*2.^0+1*2.^1+0*2.^2+0*2.^3+1*2.^4+0*2.^5+0*2.^6
ans = 18
>> 0*2.^0+1*2.^1+0*2.^2+0*2.^3+1*2.^4+0*2.^5+1*2.^6
ans = 82
```

Покажем, что десятичные числа x2.data совпадают с рассчитанными по формуле (1):

```
>> 18*slope+bias
ans = 0.5625
>> 82*slope+bias
ans = 2.5625
```

Добавим параметр bias = 5 при том же значении slope = 2^(–5):

```
>> a=[0.5555555 2.5555555];
>> bias=5;
>> slope=2^(–5);
>> x3=fi(a,1,8,slope,bias);
>> x3.data
ans = 1.0000        2.5625
>> x3.bin
ans = 10000000      10110010
>> x3.int
ans = –128          –78
```

Покажем, что десятичные числа x3.data совпадают с рассчитанными по формуле (1):

```
>> (–128)*slope+bias
ans = 1
>> (–78)*slope+bias
ans = 2.5625
```

Изменение данных с ФТ с помощью параметров slope и bias называется масштабированием данных с ФТ. Его не следует путать ни с нормированием исходных неквантованных данных, ни масштабированием с целью минимизации переполнений.

Формирование воздействия с ФТ

Сформировать воздействие с квантованными значениями, моделирующее цифровой сигнал с ФТ на выходе АЦП, можно одним из следующих двух способов:

- на основе объекта quantizer с помощью функции quantize;
- на основе объекта fi.

Отметим, что если исходные неквантованные значения воздействия по модулю превосходят единицу, то предварительно их следует нормировать, а далее при вычислении реакции учитывать нормирующий множитель.

Пример 10

Сформировать воздействие xq1, моделирующее цифровой сигнал с ФТ на выходе 12-разрядного АЦП, на основе объекта quantizer с помощью функции quantize. В качестве исходного, дискретного, сигнала x_in с неквантованными значениями (числами типа double) использовать равномерный белый шум, генерируемый с помощью функции rand(100,1), значения которой не превосходят единицу и потому не требуют нормирования. Сохранить векторы x_in и xq1 на диске.

Для формирования воздействия xq1 выполним следующие действия:

1. Создадим и сохраним на диске объект quantizer с именем q1, моделирующий 12-разрядное АЦП, со следующими свойствами (пример 1):

```
>> q1=quantizer('Mode','fixed','RoundMode','convergent',...
'OverflowMode','saturate','Format',[12 11])
q1 =
            DataMode  =  fixed
           RoundMode  =  convergent
        OverflowMode  =  saturate
              Format  =  [12 11]
>> save q1
```

2. Создадим вектор x_in неквантованных значений — дискретный сигнал, который в данном случае представляет собой равномерный белый шум, и сохраним его на диске:

```
>> x_in=rand(100,1);
>> save x_in
```

3. С помощью функции quantize создадим вектор xq1 квантованных значений с ФТ — цифровой сигнал на выходе 12-разрядного АЦП — и сохраним его на диске:

```
>> xq1=quantize(q1,x_in);
>> save xq1
```

Пример 11

Сформировать воздействие xq2, моделирующее цифровой сигнал с ФТ на выходе 12-разрядного АЦП на основе объекта fi. В качестве исходного дискретного сигнала с неквантованными значениями использовать сохраненный на диске равномерный белый шум x_in (пример 10).

Для формирования воздействия xq2 выполним следующие действия:

1. Создадим вектор x_in неквантованных значений — дискретный сигнал, в данном случае представляющий собой равномерный белый шум x_in:

```
>> load x_in
```

2. Создадим объект fi с именем xq2 и входными параметрами w = 12, f = 11, s = 1 и выполним квантование в векторе x_in:

```
>> xq2=fi(x_in,1,12,11);
```

Квантованные значения объекта xq2 хранятся в поле data, поэтому далее используется вектор xq2.data.

3. Установим требуемые свойства объекта xq2. Вывести полный список свойств объекта xq2 по команде get(xq2) и установить их требуемые значения.

Из соображений экономии места список свойств не приводится. В данном случае изменим только значение свойства RoundMode, а остальные установим по умолчанию (они аналогичны свойствам объекта в примере 10):

```
>> xq2.RoundMode='convergent';
```

4. Сохраним объект xq2 с новыми свойствами на диске:

```
>> save xq2
```

Критерии сравнения квантованного сигнала с неквантованным

Для сравнения векторов с квантованными и неквантованными элементами (квантованных и дискретных[1] сигналов) в качестве объективных критериев обычно используют нормы:

$$\|a-b\|_1 = \sum_{i=1}^{m} a|a_i-b_i|; \qquad (3)$$

$$\|a-b\|_\infty = \max_i |a_i-b_i|; \qquad (4)$$

$$\|a-b\|_2 = \sqrt{\sum_{i=1}^{m} |a_i-b_i|^2}, \qquad (5)$$

где a, b — сравниваемые векторы (квантованный и дискретный сигналы); i — номер элемента; m — количество элементов.

[1] Условно это последовательность чисел с ПТ типа double.

Пример 12

Сравнить векторы x_in и xq2.data на основе норм (3–5), используя функцию norm:

```
>> norm((x_in–xq2.data),1)
ans = 0.0121
>> norm(x_in–xq2.data)
ans = 0.0014
>> norm((x_in–xq2.data),inf)
ans = 2.4133e–004
```

Полученные значения свидетельствуют о близости векторов.

Вычисление реакции ЦФ с ФТ

Для вычисления реакции ЦФ с ФТ, описанных в виде объектов dfilt, используется функция filter, простейший формат которой при нулевых начальных условиях имеет вид:

```
y=filter(Hd,x)
```

Здесь Hd — объект dfilt, описывающий КИХ- или БИХ-фильтр с ФТ; x, y — воздействие и реакция.

Реакция КИХ-фильтров с ФТ

Вычисление реакции КИХ-фильтров с ФТ поясним на примере. Воздействие сформируем на основе объекта quantizer с помощью функции quantize (пример 10).

Пример 13

Необходимо:

- вычислить реакцию y_out исходного КИХ-фильтра (объекта Hf3, пример 1 в [9]) на воздействие, заданное вектором x_in неквантованных значений (пример 10);
- вычислить реакцию yq_out КИХ-фильтра с ФТ с 16-разрядными коэффициентами (объекта Hq3c1, пример 5 в [9]) на воздействие, заданное вектором xq1 квантованных 12-разрядных значений (пример 10);
- сравнить реакции y_out и yq_out на основе норм (3–5).

Для решения задачи выполним следующие действия:

1. Загрузим сохраненный КИХ-фильтр с ФТ (объект Hq3c1) и выведем список его свойств (для экономии места он не приводится, смысл свойств поясняется в [5]):

```
>> load Hq3c1
>> get(Hq3c1)
```

Изменим значения требуемых свойств объекта Hq3c1 и сохраним объект Hq3c1 с новыми свойствами на диске:

```
>> set(Hq3c1,'OverflowMode','saturate')
>> set(Hq3c1,'ProductMode','KeepMSB')
>> set(Hq3c1,'OutputMode','SpecifyPrecision')
>> set(Hq3c1,'OutputFracLength',15)
>> save Hq3c1
```

Если пользователь ориентируется на реализацию фильтра с ФТ на базе конкретного ЦПОС, то соответствующие значения свойств должны быть согласованы с характеристиками ЦПОС. В данном случае значения свойств выбраны из соображений разумной логики.

2. Вычислим реакции y_out и yq_out с помощью функции filter:

```
>> load Hf3
>> load x_in
>> y_out=filter(Hf3,x_in);
>> load Hq3c1
>> load xq1
>> yq_out=filter(Hq3c1,xq1);
```

После загрузки объектов dfilt в рабочее пространство **Workspace** рекомендуется просмотреть, какие переменные для объекта dfilt были использованы по умолчанию, во избежание их дублирования.

Реакция y_out исходного КИХ-фильтра на воздействие x_in представляет собой вектор неквантованных значений. Длины векторов x_in и y_out совпадают.

Реакция yq_out КИХ-фильтра с ФТ представляет собой объект fi. Квантованные значения объекта yq_out хранятся в поле data, поэтому далее используется вектор yq_out.data. Длины векторов xq1 и yq_out.data совпадают.

3. Сравним векторы y_out и yq_out.data на основе норм (3–5):

```
>> norm((y_out–yq_out.data),1)
ans = 0.0056
>> norm(y_out–yq_out.data)
ans = 7.2107e–004
>> norm((y_out–yq_out.data),inf)
ans = 2.1064e–004
```

Полученные значения свидетельствуют о близости векторов.

Реакция БИХ-фильтров с ФТ

Вычисление реакции БИХ-фильтров с ФТ поясним на примере. Воздействие сформируем на основе объекта fi (пример 10).

Пример 14

Необходимо:

- вычислить реакцию Y_OUT исходного БИХ-фильтра (объекта Hn, пример 4 в [10]) на воздействие, заданное вектором x_in неквантованных значений (пример 11);
- вычислить реакцию YQ_OUT БИХ-фильтра с ФТ с 16-разрядными коэффициентами (объекта Hqn2, пример 5 в [10]) на воздействие, заданное вектором xq2 квантованных 12-разрядных значений (пример 11);
- сравнить реакции Y_OUT и YQ_OUT на основе норм (3–5).

Для решения задачи выполним следующие действия:

1. Загрузим сохраненный БИХ-фильтр с ФТ (объект Hqn2) и выведем список его свойств (для экономии места он не приводится, смысл свойств поясняется в [5]):

```
>> load Hqn2
>> get(Hqn2)
```

Изменим значения требуемых свойств объекта Hqn2 и сохраним созданный объект Hqn2 с новыми свойствами на диске:

```
>> set(Hqn2,'OverflowMode','saturate')
>> set(Hqn2,'ProductMode','KeepMSB')
>> set(Hqn2,'OutputMode','SpecifyPrecision')
>> set(Hqn2,'OutputFracLength',15)
>> set(Hqn2,'StageInputAutoScale',0)
>> set(Hqn2,'StageInputFracLength',15)
>> set(Hqn2,'StageOutputAutoScale',0)
>> set(Hqn2,'StageOutputFracLength',15)
>> save Hqn2
```

2. Вычислим реакции Y_OUT и YQ_OUT с помощью функции filter:

```
>> load Hn
>> load x_in
>> Y_OUT=filter(Hn,x_in);
>> load xq2
>> load Hqn2
>> YQ_OUT=filter(Hqn2,xq2);
```

Реакция Y_OUT исходного БИХ-фильтра на воздействие x_in представляет собой вектор неквантованных значений. Длины векторов x_in и Y_OUT совпадают.

Реакция YQ_OUT БИХ-фильтра с ФТ представляет собой объект fi. Квантованные значения объекта YQ_OUT хранятся в поле data, поэтому далее используется вектор YQ_OUT.data. Длины векторов xq2 и YQ_OUT.data совпадают.

3. Сравним векторы Y и Y1.data на основе норм (3–5):

```
>> norm((Y_OUT–YQ_OUT.data),1)
ans = 0.0177
>> norm(Y_OUT–YQ_OUT.data)
ans = 0.0023
>> norm((Y_OUT–YQ_OUT.data),inf)
ans = 5.6329e–004
```

Полученные значения свидетельствуют о близости векторов.

Литература

1. Ingle V., Proakis J. Digital Signal Processing Using MATLAB. Second Edition. Thomson.
2. Оппенгейм А., Шафер Р. Цифровая обработка сигналов. М.: Техносфера, 2006.
3. Сергиенко А. Б. Цифровая обработка сигналов, 2-е изд. СПб.: ПИТЕР, 2006.
4. Солонина А. И., Улахович Д. А., Арбузов С. М., Соловьева Е. Б. Основы цифровой обработки сигналов. 2-е изд. СПб.: БХВ-Петербург, 2005.
5. Солонина А. И., Арбузов С. М. Цифровая обработка сигналов. Моделирование в MATLAB. СПб.: БХВ-Петербург, 2008.
6. Солонина А. Моделирование цифровой обработки сигналов в MATLAB. Часть 1. Синтез оп-

тимальных (по Чебышеву) КИХ-фильтров программными средствами MATLAB // Компоненты и технологии. 2008. № 11.
7. Солонина А. Моделирование цифровой обработки сигналов в MATLAB. Часть 2. Синтез оптимальных БИХ-фильтров программными средствами MATLAB // Компоненты и технологии. 2008. № 12.
8. Солонина А. Моделирование цифровой обработки сигналов в MATLAB. Часть 3. Описание структур КИХ- и БИХ-фильтров в MATLAB // Компоненты и технологии. 2009. № 1.
9. Солонина А. Моделирование цифровой обработки сигналов в MATLAB. Часть 4. Моделирование структур цифровых фильтров с фиксированной точкой программными средствами MATLAB: анализ характеристик КИХ-фильтров // Компоненты и технологии. 2009. № 2.
10. Солонина А. Моделирование цифровой обработки сигналов в MATLAB. Часть 5. Моделирование структур цифровых фильтров с фиксированной точкой программными средствами MATLAB: анализ характеристик БИХ-фильтров // Компоненты и технологии. 2009. № 3.