第7章数字信号处理中的有限字长效应

BACK

7.1 数值表示的有限字长效应

7.1.1 定点数与浮点数

为了用有限的数字符号代表所有的数值，人们通常采用进位制（也称进制）的方法，即按给定的规则进位。例如，对于R进制，就表示某一位置上的数运算时每逢R进一位。R进制下，任何一个数P可以表示为

$$(P)_R = \sum_{i=m}^{n} k_i R^i \tag{7-1}$$

当式（7-1）中的下限$m \geq 0$时，P是纯整数，上限$n \leq 0$时，P是纯小数。可以看出，$i=0$ 是区分数P的整数部分和小数部分的关键位置，称为小数点位置。

根据小数点位置是否变化，将数的表示区分为定点数和浮点数两种。小数点位置固定不变的数称为定点数，如11.556，3.189等。定点数表示数的格式规范，他所表示的数的精度固定。小数点位置可以变化的数称为浮点数。浮点数所表示数的精度是不同的，如 111.76， 1.2357，1000.1等。同一个数也可以用不同的形式来表示，如111.76，可以用科学计数法表示成11.176×10^{1}，1.1176×10^{2}，0.11176×10^{3}等。

数字系统中，所采用的二进制表示法有定点制和浮点制两种。无论是采用软件编程还是采用专用硬件实现，存储和处理过程中所采用的存储单元的长度都只能是有限的，也就是说数值表达的精度是有限的。接下来对二进制的定点、浮点表示因寄存器长度限制而导致的误差进行分析，并就折中的成组浮点制（Block Floating Point，BFP，也称块浮点制）进行简单讨论。

7.1.2 定点制误差分析

1. 数的定点表示

定点制下，一旦确定了小数点在整个数码中的位置，在整个运算过程中即保持不变。因此，根据系统设计要求、数值范围来确定小数点处于什么位置很重要，这就是数的定标。

数的定标有Q表示法和S表示法两种。Q表示法形如Q*n*，字母Q后的数值n表示包含n位小数。如Q0表示小数点在第0位的后面，数为整数；Q15 表示小数点在第15位的后面，0～14位都是小数位。S表示法则形如S*m*.*n*，*m*表示整数位，*n*表示小数位。以16位DSP为例，通过设定小数点在16位数中的不同位置，可以表示不同大小和不同精度的小数。表7.1列出了一个16位数的16种Q表示、S表示及它们所能表示的十进制数值范围。

表 7.1　Q 表示、S 表示及数值范围

Q 表示	S 表示	十进制数表示范围	分辨率
Q15	S0.15	$-1 \leqslant X \leqslant 0.9999695$	$2^{-15}=0.000030517578125$
Q14	S1.14	$-2 \leqslant X \leqslant 1.9999390$	$2^{-14}=0.00006103515625$
Q13	S2.13	$-4 \leqslant X \leqslant 3.9998779$	$2^{-13}=0.0001220703125$
Q12	S3.12	$-8 \leqslant X \leqslant 7.9997559$	$2^{-12}=0.000244140625$
Q11	S4.11	$-16 \leqslant X \leqslant 15.9995117$	$2^{-11}=0.00048828125$
Q10	S5.10	$-32 \leqslant X \leqslant 31.9990234$	$2^{-10}=0.0009765625$
Q9	S6.9	$-64 \leqslant X \leqslant 63.9980469$	$2^{-9}=0.001953125$
Q8	S7.8	$-128 \leqslant X \leqslant 127.9960938$	$2^{-8}=0.00390625$
Q7	S8.7	$-256 \leqslant X \leqslant 255.9921875$	$2^{-7}=0.0078125$
Q6	S9.6	$-512 \leqslant X \leqslant 511.9804375$	$2^{-6}=0.015625$
Q5	S10.5	$-1024 \leqslant X \leqslant 1023.96875$	$2^{-5}=0.03125$
Q4	S11.4	$-2048 \leqslant X \leqslant 2047.9375$	$2^{-4}=0.0625$
Q3	S12.3	$-4096 \leqslant X \leqslant 4095.875$	$2^{-3}=0.125$
Q2	S13.2	$-8192 \leqslant X \leqslant 8191.75$	$2^{-2}=0.25$
Q1	S14.1	$-16384 \leqslant X \leqslant 16383.5$	$2^{-1}=0.5$
Q0	S15.0	$-32768 \leqslant X \leqslant 32767$	$2^{0}=1$

2. 定点运算

定点表示的两个数在进行加减法运算前，必须保证这两个数的定标值Q严格相等。假设进行加减运算的两个数分别为x和y，它们的Q值相等且都为Q_c，则进行加减运算的结果为$x \pm y$，结果的Q值仍为Q_c。

当x和y的Q值不相等时，假设它们的Q值分别为Q_x和Q_y，且有$Q_x > Q_y$，运算结果z的定标值为Q_z，进行加减运算的步骤如下：

(1) 将Q值较小的数y的Q值调整为Q_x，即$y' = y \times 2^{(Q_x - Q_y)}$；

(2) 计算$z' = x \pm y'$的值，其Q值为Q_x；

(3) 将z'的Q值调整到Q_z，即：$z = z' \times 2^{(Q_z - Q_x)}$，得到最终的运算结果。

乘除运算时，假设进行运算的两个数分别为x和y，它们的Q值分别为Q_x和Q_y，则两者进行乘法运算的结果为xy，Q值为Q_x+Q_y，除法运算的结果为x/y，Q值为Q_x-Q_y。

在程序或硬件实现中，上述定标值的调整可以直接通过寄存器的左移或右移完成。若$b>0$，实现$x\times 2^b$需将存储x的寄存器左移b位；若$b<0$，实现$x\times 2^b$则需将存储x的寄存器右移$|b|$位即可。

3. 定点数的原码、补码和反码表示

从前面的分析过程可知，采用定点处理器进行小数的运算，其小数点位是由程序员根据处理问题的需要人为设定的，在处理的过程中需要时刻关注Q值的变化及其所表示的物理含义，因而，从处理器的角度来看，定点制下进行小数的运算实质上还是按处理器字长进行整数的各种运算。同样，对数进行存储和处理的时候，数的各种码制表示规律也同样适用于定点表示下的小数处理。

计算机基础课程中，计算机在表达数值的时候，二进制数的最高位一般用作符号位，0表示正数，1表示负数，一个数可以有原码、补码和反码三种形式。对于正数来说，三种码都一样，而对于负数，这三种码并不一致，用途也各不相同。如原码适合做乘除法，常被用作设计串行乘法器，而补码适合做加减法，加法器的硬件多采用补码制。表7.2列出了三种码的特性对比。在数字信号处理器（DSP）中，通常采用补码制。

表 7.2　原码、补码和反码特性对比表

类　别	原 码	补 码	反 码
定　义	“符号一幅度码”，最高位为符号位，代表数的正负，其余位代表数的大小	正数与原码表示相同，负数“取反加一”	正数与原码表示相同，负数按位取反
表示数的数量	2^n-1（0 有两种表示，即 $(00..0)_2$ 和 $(10..0)_2$）	2^n（0 仅有一种表示，即 $(00..0)_2$）	2^n-1（0 有两种表示，即 $(00..0)_2$ 和 $(11..1)_2$）
优　点	乘除运算方便，以两数符号位的逻辑加就可简单决定结果的正负号，数值则是两数数值部分乘除的结果	（1）求取方便，负数直接取反加一即得其补码； （2）可以把减法与加法统一成补码加法，符号位同时参加运算，无需考虑溢出进位； （3）只要最后相加结果不溢出，即使中间结果有溢出也不影响运算	（1）求取方便，负数直接按位取反即得反码； （2）也可以把减法与加法统一成加法运算，但如果符号位相加后出现进位，则需把它送回到最低位进行相加，即做循环移位与最低位相加
缺　点	加减运算不方便，需先判断符号是否相同，相同则做加法，不同则做减法，减法时还需要判断两数绝对值大小，用绝对值大者减绝对值小者，增加了运算开销	乘除运算不方便	乘除运算不方便
应用场合	较多使用，典型应用为串行乘法器	较多使用，程序处理和硬件加法器设计中常见	不常用
举　例	$(0.111)_2=(0.875)_{10}$ $(1.111)_2=(-0.875)_{10}$	$(0.111)_2=(0.875)_{10}$ $(1.111)_2=(-0.125)_{10}$	$(0.111)_2=(0.875)_{10}$ $(1.111)_2=(0)_{10}$

数的转换在MATLAB中的实现有如下几种方式。

(1) 函数dec2bin可以实现将一个十进制正整数转换成一个二进制的字符串。 调用格式：

dec2bin(D)

dec2bin(D,N)

输入变量：D是小于2^{52}的非负整数，N为转换后的二进制字符串的长度。

(2) 函数bitcmp 可以用于求一个十进制正整数的补数，和函数dec2bin一起使用可以求一个十进制整数的反码和补码。调用格式：

bitcmp(A,N)

输入变量：A是无符号型整数，其中$A \leq 2^N-1$，计算结果为2^N-1-A。

(3) 函数num2bin用于将一个数值矩阵转化成二进制的字符串。 调用格式：

B = num2bin(Q,X)

输入变量：*X*是数值矩阵，Q用于表明*X*的属性。在MATLAB里用函数?quantizer生成量化目标Q，常用的调用格式为Q=quantizer(［w,f］)，*w*是字符串B的长度，*f*是小数位数。需要注意的是，如果不是0.xxxx，必须要给整数位保留两个比特（包含一位符号位）。

【例7-1】 用MATLAB编程求十进制数 -1325的二进制原码、反码和补码表示。

解： 因为2^10<1325<2^11，需要用11位的二进制数来表示该十进制数，假设用12位二进制数表示，其中第12位为符号位。利用前面介绍的MATLAB函数编写程序如下：

```
x=-1325;
a=abs(x);
b=dec2bin(a+2^11,12)                    %原码
c=dec2bin(bitcmp(a,12),12)              %反码
d= dec2bin(bitcmp(a,12)+1,12)           %补码
```

运行结果：

```
b=110100101101
c=101011010010
d=101011010011
```

■

【例7-2】 用MATLAB编程求 十进制数123.874的二进制原码表示，小数位数保留8位。

解： 利用前面介绍的MATLAB函数编写程序如下：

```
Q=quantizer(［16,8］);
num2bin(Q,123.874)
```

运行结果：

```
ans=
    0111101111011111
```

■

4. 定点量化误差分析

定点制下，当Q值确定以后，即用来表示小数的寄存器位数L确定后，其可表示的最小数的单位也就确定了，记为2^{-L}，这个值称为量化间距，记作q。超出L位的部分，可以通过直接截断的方式进行处理，所产生的误差称为截尾误差，也可以通过舍入的方式进行处理，产生舍入误差。

如果数P的小数部分是x，通过M位二进制表示，存入Q值定义为L的寄存器中被量化为$Q[x]$，则其量化误差e定义为

$$e=Q[x]-xx\in[0,1) \tag{7-2}$$

1） 截尾误差

对于正数，原码、 补码和反码的形式都相同，有

$$x=\sum_{i=1}^{M}k_i2^{-i} \tag{7-3}$$

$$Q[x]=\sum_{i=1}^{L}k_i2^{-i} \tag{7-4}$$

$$e=Q[x]-x=-\sum_{i=L+1}^{M}k_i2^{-i} \tag{7-5}$$

显然，此时有$-(2^{-L}-2^{-M})\le e\le 0$，当$M\to\infty$时，有$-2^{-L}\le e\le 0$。也就是说，截尾误差最大不超出量化间距$q$。

对于负数，由于三种码的表达方式不同，误差也不同。

$$x = -\sum_{i=1}^{M} k_i 2^{-i} \tag{7-6}$$

$$Q[x] = -\sum_{i=1}^{L} k_i 2^{-i} \tag{7-7}$$

$$e = Q[x] - x = \sum_{i=L+1}^{M} k_i 2^{-i} \tag{7-8}$$

易知，$0 \le e \le 2^{-L} - 2^{-M}$。当$M \to \infty$时，有$0 \le e \le 2^{-L}$。也就是说，用原码表示负数时，截尾误差始终为正，误差均值为$2^{-(L+1)}$，且最大误差不超出量化间距q。

当负数用补码表示时，有

$$x = -1 + \sum_{i=1}^{M} k_i 2^{-i} \tag{7-9}$$

$$Q[x] = -1 + \sum_{i=1}^{L} k_i 2^{-i} \tag{7-10}$$

$$e = Q[x] - x = \quad -\sum_{i=L+1}^{M} k_i 2^{-i} \tag{7-11}$$

易知，$-(2^{-L}-2^{-M}) \le e \le 0$。当$M \to \infty$时，有$-2^{-L} \le e \le 0$。也就是说，用补码表示负数时，其截尾误差与正数情况一致，始终为负，均值为$-2^{-(L+1)}$，最大误差不超出量化间距q。当负数用反码表示时，有

$$x = -1 + \sum_{i=1}^{M} k_i 2^{-i} + 2^{-M} \tag{7-12}$$

$$Q[x] = -1 + \sum_{i=1}^{L} k_i 2^{-i} + 2^{-L} \tag{7-13}$$

$$e = Q[x] - x = -\sum_{i=L+1}^{M} k_i 2^{-i} + 2^{-L} - 2^{-M} \tag{7-14}$$

2）舍入误差

当定点数用于表示小数的寄存器长度为L，将长度为$M(M>L)$的数据存入该寄存器进行舍入处理时，就是将第$L+1$位加1，然后再截断数据到第L位。容易理解，无论采用原码、补码还是反码表示，这个过程都调整了误差范围，误差均值的绝对值由$2^{-(L+1)}$调整为0，而误差范围调整为［$-2^{-(L+1)}$，$2^{-(L+1)}$］。

7.1.3 浮点制误差分析

从前面对定点制表示分析的过程可以看出，定点制的优点是运算简便，对处理器要求低。但数的表达能力有限，所处理数的动态范围较小。同时，寄存器的利用效率低，如表示较小的小数时，小数的有效位数较短，由截尾舍入产生的百分比误差随着数的绝对值的减小而增加。

科学计数法中，允许将任意一个数字表示为一个纯小数与一个指数相乘的形式

$$(P)_R=S\cdot R^c$$

S一般为绝对值介于0~1之间的规格化尾数，机器中可用原码或补码表示。R为基数，在数字处理器中通常取2。c为浮点数的阶码，也即是指数，为整数。采用浮点制表示，可以在兼顾表达范围的同时保证运算精度。

根据IEEE754国际标准，常用的浮点数有单精度浮点数(single)和双精度浮点数(double)两种格式。单精度浮点数占用4个字节(32位)存储空间，在数字寄存器中，单精度浮点数的存储格式如图7-1 所示，包括1位符号位S，8位阶码(指数)E，23位尾数F。 其表示范围为

(1) 当S=0，E=127，F的23位均为1时，表示的浮点数为最大的正数：

$$(01.111..1)_2\times 2^{127}=(2-2^{-23})\times 2^{127}\approx 3.4\times 10^{38}$$

(2) 当S=1，E=127，F的23位均为0时，表示的浮点数为绝对值最大的负数：

$$(10.000..0)_2\times 2^{-127}=-2\times 2^{-127}\approx -3.4\times 10^{38}$$

(3) 当S=0，E=－127，F的23位均为0时，表示的浮点数为最小的正数：

$$(01.000..0)_2\times2^{-127}=1\times2^{-127}\approx5.9\times10^{-39}$$

(4) 当S=1，E=－127，F的23位均为1时，表示的浮点数为绝对值最小的负数：

$$(10.111..1)_2\times2^{-127}=(-1-2^{-23})\times2^{-127}\approx-5.9\times10^{-39}$$

双精度浮点数占用8个字节(64位)存储空间，包括1位符号位、 11位阶码、 52位尾数，数值范围为1.7E-308～1.7E+308。

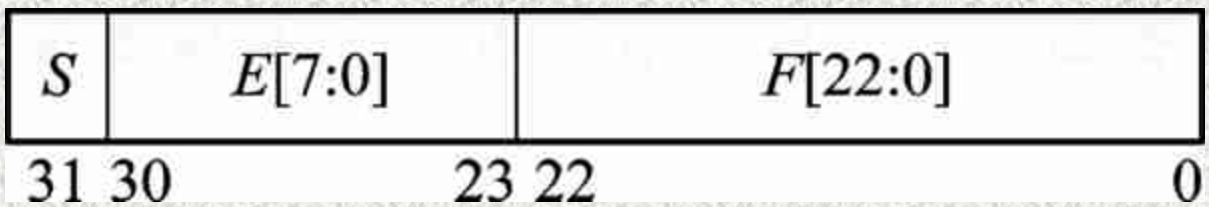

图7-1 单精度浮点数存储格式

浮点数进行加减运算一般需要有五个步骤：

(1) 对阶： 使两数的小数点位置对齐。

(2) 尾数求和(差)： 将对阶后的两尾数按定点加减运算规则求和(差)。

(3) 规格化: 为增加有效数字的位数，提高运算精度，必须将求和(差)后的尾数规格化。

(4) 舍入： 为提高精度，要考虑尾数右移时丢失的数值位。

(5) 判断结果： 即判断结果是否溢出。

进行浮点数的乘除运算分为阶码运算和尾数运算两个步骤。阶码运算较为简便，若是乘法则对阶码寄存器段做加法运算，除法则做减法运算。

对于浮点数的乘法，尾数运算一般需要四个步骤：

(1) 预处理：检测两个尾数中是否有一个为0，若有一个为0，乘积必为0，不再作其他操作；如果两尾数均不为0，则可进行乘法运算。

(2) 相乘：两个浮点数的尾数相乘可以采用定点小数的任何一种乘法运算来完成。

(3) 规格化：相乘结果可能要进行左规，左规时调整阶码后如果发生阶下溢，则作机器零处理；如果发生阶上溢，则作溢出处理。

(4) 尾数截断：尾数相乘会得到一个双倍字长的结果，若限定只取1倍字长，则乘积的若干低位将会丢失。可以通过前面讨论的定点制截断处理方法进行截尾或舍入处理。

对于浮点除法，尾数运算需要先检测被除数是否为0，若为0，则商为0；再检测除数是否为0，若为0，则商为无穷大，另作处理。若两数均不为0，则可进行除法运算。两浮点数尾数相除同样可采取定点小数的任何一种除法运算来完成。对已规格化的尾数，为了防止除法运算结果溢出，可先比较被除数和除数的绝对值，如果被除数的绝对值大于除数的绝对值，则先将被除数右移一位，其阶码加1，再作尾数相除。此时所得结果必然是规格化的定点小数。

与定点制相比，浮点制一定程度上可以兼顾动态范围和运算精度，但其指数和尾数都需要参与运算，运算过程复杂，实现难度大，硬件成本高，在仅提供定点运算的处理器中很难获得实时的运算结果。

从处理误差方面看，浮点制所产生的误差传播范围广，分析起来难度较大，此处不做详细论述，可参阅相关书籍。

7.1.4 成组浮点制(BFP)

成组浮点制(Block Floating Point)也称为块浮点制，基本思想是兼顾定点制和浮点制的优点，将一组数值相近的数据定义成一个具有统一指数的数据块，换句话说，就是将该组数据同时根据这个共享指数进行缩放，在保证动态范围和精度的同时又不需要考虑彼此间指数的影响。

BFP是一种在工程实现中较常采用的方法。如Altra公司提供的IP核FFT Megacore中就集成了该算法。

7.2 A/D变换的有限字长效应

7.2.1 A/D变换及其量化误差的统计分析

A/D变换包括采样和量化两部分，如图7-2所示。模拟信号$x_a(t)$经过采样后转换为时域离散信号$x(n)$，然后对$x(n)$做截尾或者舍入的量化处理后得到二进制数字信号$\hat{x}(n)$。

由于A/D变换总是采用定点制，因此需要将模拟信号乘以比例因子A，以限定其最大值不能超过A/D变换的动态范围，即

$$x(n)=Ax_a(t)|_{t=nT}=Ax_a(nT) \tag{7-15}$$

显然信号$x(n)$具有无限精度，受存储单元的字长限制，必须对其做截尾或者舍入的量化处理，用$e(n)$表示量化误差，则

$$e(n)=Q[x(n)]-x(n)=\hat{x}(n)-x(n) \tag{7-16}$$

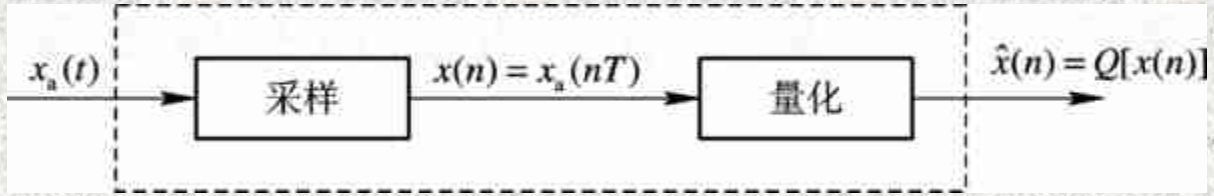

图7-2 A/D变换器的非线性模型

A/D变换的量化方式和数的表示方式直接决定了其量化特性。设A/D变换的输出是字长为b+1位的补码定点小数，其中b位是小数部分，若用$e_T(n)$表示采用截尾的误差，用$e_R(n)$表示采用舍入处理的误差，由7.1节可以得到量化误差的范围为

$$-q < \mathrm{e}_T(n) \le 0 \tag{7-17}$$

$$-\frac{q}{2} < \mathrm{e}_R(n) \le \frac{q}{2} \tag{7-18}$$

这里，$q=2^{-b}$表示量化阶距。式(7-17)、(7-18)给出了量化误差的范围，但是要想精确地刻画出每个量化误差的值还是非常困难的。通常用统计分析的方法来分析量化误差的统计特性，研究A/D变换的有限字长效应。A/D变换器的统计模型如图7-3所示。

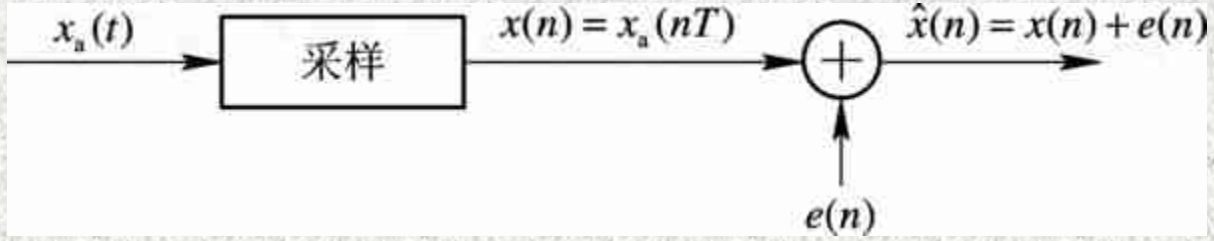

图7-3 A/D变换器的统计模型

为了研究其统计特性，首先对量化误差$e(n)$作如下假设：

(1) $e(n)$是平稳随机序列；

(2) $e(n)$与采样信号$x(n)$互不相关；

(3) $e(n)$序列中任意两个值之间不相关，即$e(n)$为白噪声序列；

(4) $e(n)$在其取值范围内均匀等概分布。

根据上述假设， $e(n)$是与输入信号完全不相关的、 均匀分布的白噪声序列，采用截尾和舍入时误差的概率密度函数分别如图7-4(a)、 (b)所示。

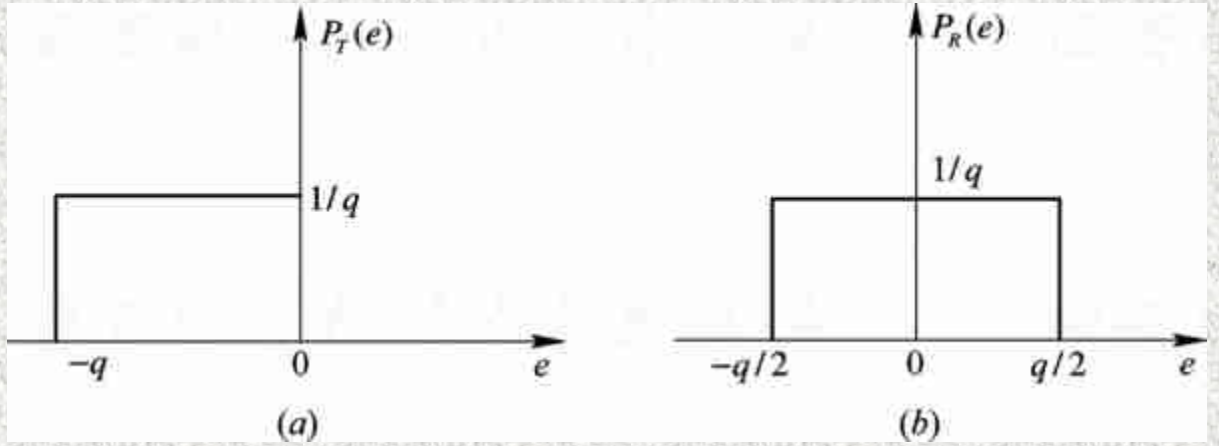

图7-4 量化误差的概率分布

(a) 截尾误差；(b) 舍入误差

采用截尾处理时的均值和方差分别为

$$m_T = \int_{-\infty}^{+\infty} e_T P_T(\mathrm{e}_T) de_T = \int_{-q}^{0} \frac{1}{q} e_T de_T = -\frac{q}{2} = -2^{-b-1} \tag{7-19}$$

$$\sigma_T^2 = \int_{-\infty}^{+\infty} (e_T - m_T)^2 P_T(\mathrm{e}_T) de_T = \int_{-q}^{0} (e_T + \frac{q}{2})^2 \frac{1}{q} de_T = \frac{q^2}{12} = \frac{1}{12} 2^{-2b} \tag{7-20}$$

采用舍入处理时的均值和方差分别为

$$m_R = \int_{-\infty}^{+\infty} e_R P_R(\mathrm{e}_R) de_R = \int_{-q/2}^{q/2} \frac{1}{q} e_R de_R = 0 \tag{7-21}$$

$$\sigma_R^2 = \int_{-\infty}^{+\infty} (e_R - m_R)^2 P_R(\mathrm{e}_R) de_R = \int_{-q/2}^{q/2} e_R{}^2 \frac{1}{q} de_R = \frac{q^2}{12} = \frac{1}{12} 2^{-2b} \tag{7-22}$$

分析式(7-19)~(7-22)可知：

(1) 采用截尾处理时误差序列的均值不为零，也就是说误差序列$e_T(n)$中包含直流分量，直流分量的存在会使信号的频谱在频率等于0处存在δ函数，从而影响信号的频谱结构。而采用舍入处理时误差序列的均值为0，不存在直流分量。因此，实际应用中多采用舍入处理的方式。

(2) 采用截尾处理和舍入处理时误差序列的方差相等，且都为$\sigma^2_T=\sigma^2_R=\sigma^2_e$，都取决于A/D变换的字长$b$，字长越长，量化误差越小。

用符号σ^2_x表示信号功率，则量化信噪比为

$$\frac{\sigma_x^2}{\sigma_e^2}=\frac{\sigma_x^2}{q^2/12}=(12\times 2^{2b})\sigma_x^2 \tag{7-23}$$

对数表示为

$$SNR = 10\lg\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10\lg\left[(12\times 2^{2b})\sigma_x^2\right] \tag{7-24}$$
$$= 6.02(b+1) + 10\lg(3\sigma_x^2)$$

从式(7-24)可以看出，信号功率一定的情况下，字长每增加1 bit，量化信噪比约增加6个分贝。

【例7-3】 假设语音信号量化编码时，选用12 bit 的A/D，其动态范围为0~5 V，求系统量化误差的均方差。

解： 量化阶距电压：

$$V_q = 5\times 2^{-b} = 5\times 2^{-12} = 1.2\ \text{mV}$$

$$\sigma_e = \frac{V_q}{\sqrt{12}} = 0.3464\,\text{mV}$$