东北大学秦皇岛分校

计算机与通信工程学院

Matlab 与通信系统仿真

结课作业

所在班级：通信 2004 班

学生学号：202012544

学生姓名：韩泽华

指导教师：刘福来

报告成绩：

评　　语：

基于算术编码的信源编/解码通信系统设计与仿真

前 言

我们已经进入了数字化信息的时代，各种媒体如数值、文字、语音、图像等都要转换成计算机能够处理的数字形式，但这样会产生大量的数据。这对存储器的容量、通信干线的带宽和计算机的速度都提出了很高的挑战。要解决这个问题，不能仅靠增加存储空间和传输速率，必须采用数据压缩技术来减少信息数据量，从而节省存储空间和提高传输效率。因此数据压缩技术越来越受到重视，从基本的无损压缩到语音、图像和视频等信号的有损压缩，数据压缩在我们的日常生活中越来越发挥着重要作用。

根据解码后数据与原始数据是否完全一致进行分类，数据压缩方法一般分为两类：①无损压缩，即解码图像与原始图像严格相同，压缩比大约在 2：1－5：1 之间，如霍夫曼编码，算术编码等；②有损编码，即还原图像与原始图像存在一定的误差，但视觉效果一般可以接受，压缩比可以从几倍到上百倍，如：PCM 编码，预测编码、变换编码等。

算术编码作为一种高效的数据编码方法在文本，图像，音频等压缩中有广泛的应用。它是一种到目前为止编码效率最高的统计熵编码方法,它比著名的 Huffman 编码效率提高 10%左右。

本文主要内容：

1.介绍了信源编码的理论基础，引出离散无记忆信源的变长编码可以压缩源码的理论依据。

2.系统阐述了算数编码算法的理论基础，简要概括了算数编码的特点和分析过程，对静态算数编码与自适应算术编码进行数学建模。

3.用 MATLAB 对基于算术编码的通信系统进行设计与仿真，包括仿真流程图的设计，与最终效果的展示。

关键词：信源编码，静态算术编码，自适应算数编码，通信系统设计与仿真

目录

1 研究的目的及意义

各种媒体信息（特别是图像和动态视频）数据量非常之大。例如：一幅 640x480 分辨率的 24 位真彩色图像的数据量约力 900kb；一个 100Mb 的硬盘只能存储约 100 幅静止图像画面。显然，这样大的数据量不仅超出了计算机的存储和处理能力，更是当前通信信道的传输速率所不及的。因此，为了存储、处理和传输这些数据，必须进行压缩。相比之下，语音的数据量较小，且基本压缩方法已经成熟，目前的数据压缩研究主要集中于图像和视频信号的压缩方面。图像压缩技术、视频技术与网络技术相结合的应用前景十分可观，如远程图像传输系统、动态视频传输——可视电话、电视会议系统等已经开始商品化，MPEG 标准与视频技术相结合的产物——家用数字视盘机和 VideoCD 系统等都已进入市场。可以预计，这些技术和产品的发展将对本世纪末到二十一世纪的社会进步产生重大影响。

而算术编码作为一种高效的数据编码方法在文本，图像，音频等压缩中有广泛的应用，所以，研究算术编码以更好的利用它是非常必要的。算术编码的优点是它可以根据每个符号出现的概率动态地分配编码长度，从而使得平均编码长度接近信源熵。算术编码也可以适应不同的信源模型，包括高阶模型和自适应模型。算术编码还可以避免使用码本，减少了存储空间和传输开销。

算术编码虽然具有很多优点，但是也存在一些问题和挑战。例如，算术编码需要高精度的浮点运算，这会增加计算复杂度和硬件成本；算术编码也容易受到误差或干扰的影响，一旦出现错误或丢失数据，就可能导致整个解码过程失败；算术编码还涉及到美国专利，这会限制它在商业领域的应用。

因此，研究算术编码不仅可以提高数据压缩的效率和质量，还可以解决算术编码存在的问题和挑战，为各种媒体信息的存储、处理和传输提供更好的技术支持。

2 信源编码

2.1 信源编码的概念

信源编码是为了减少信源输出符号序列中的剩余度、提高符号的平均信息量，对信源输出的符号序列所施行的变换。具体说，就是针对信源输出符号序列的统计特性来寻找某种方法，把信源输出符号序列变换为最短的码字序列，使后者的各码元所载荷的平均信息量最大,同时又能保证无失真地恢复原来的符号序列。既然信源编码的基本目的是提高码字序列中码元的平均信息量，那么，一切旨在减少剩余度而对信源输出符号序列所施行的变换或处理，都可以在这种意义下归入信源编码的范畴，例如过滤、预测、域变换和数据压缩等。当然，这些都是广义的信源编码。

信源编码的原理是利用信息论中的概念和定理，如信息量、熵、条件熵、互信息、相对熵、数据处理不等式等，来分析和设计有效的编码方案。信源编码的方法可以分为无失真信源编码和有失真信源编码两大类。

无失真信源编码只对信源的冗余度进行压缩，不改变信源的熵，能保证码元序列无失真地恢复成信源符号的序列。无失真信源编码的基本定理是香农第一定理，它给出了无失真压缩的极限和充分必要条件。常见的无失真信源编码方法有哈夫曼编码、算术编码、游程长度编码等。

有失真信源编码也称率失真编码或熵压缩编码。它允许在解码后恢复原始符号序列时产生一定程度的失真，从而进一步降低平均码长。有失真信源编码需要在压缩率和失真之间进行权衡，通常使用率失真函数来描述这种关系。有失真信源编码的基本定理是香农第二定理，它给出了有失真压缩的极限和充分必要条件。常见的有失真信源编码方法有向量量化、变换编码、子带编码等。

2.2 信源编码的简介

信源编码是以提高通信有效性为目的的编码。通常通过压缩信源的冗余度来实现。采用的一般方法是压缩每个信源符号的平均比特数或信源的码率，同样多的信息用较少的码率来传输，使单位时间内传送的平均信息来量增加，从而提高通信的有效性。

信源编码理论是信息论的一个重要分支，其理论基础是信源编码的两个定理:无失真信源编码定理和限失真信源编码定理。前者是离散信源或数字编码的基础，

后者则是连续信源或模拟信号的基础。

编码实质上就是对信源的原始符号按一定规则进行的一种变换。编码可分为信源编码和信道编码。由于信源符号之间存在分布不均匀和相关性，使得信源存在冗余度，信源编码的主要任务就是减少冗余，提高编码效率。信源编码是为了减少信源输出符号序列中的剩余度提高符号的平均信息量，对信源输出的符号序列所施行的变换。具体说，就是针对信源输出符号序列的统计特性来寻找某种方法，把信源输出符号序列变换为最短的码字序列，使后者的各码元所载荷的平均信息量最大，同时又能保证无失真地恢复原来的符号序列。

信源编码的基本途径有两个：使序列中的各个符号尽可能地相互独立，即解除相关性；使编码中各个符号出现的概率尽可能地相等，即概率均匀化。采用的一般方法是压缩每个信源符号的平均比特数或信源的码率。即同样多的信息用较少的码率传送,使单位时间内传送的平均信息量增加,从而提高通信的有效性。

2.3 信源编码的目的：

由于信源符号之间存在概率分布不均匀和相关性，所以信源存在冗余度。信源编码的主要任务就是减少冗余，提高编码效率。信源编码是以提高通信的有效性为目的编码，通常通过压缩信源的冗余度来实现，即用较少的码字传送较多的信息，使单位时间内传送的平均信息量增加，从而提高通信的有效性。

2.4 信源编码的原理：

一般来说，减少信源输出符号序列中的剩余度、提高符号平均信息量的基本途径有两个：1、使序列中的各个符号尽可能地互相独立；2、使序列中各个符号的出现概率尽可能地相等前者称为解除相关性，后者称为概率均匀化。

信源编码的一般问题可以表述如下:

若某信源的输出为长度等于 M 的符号序列集合

$$U = \{(u_1, \dots, u_M) | u_m \in A, m = 1, \dots, M\} \quad (2\text{-}1)$$

式中符号 A 为信源符号表，它包含着 K 个不同的符号，$A = \{a_k | k = 1, \dots, K\}$，这个信源至多可以输出 K 个不同的符号序列，记 $||U|| = K$。所谓对这个信源的输出进行编码，就是用一个新的符号表 B 的符号序列集合 V 来表示信源输出的符号列集合 U。若 V 的各个序列的长度等于 N，即

$$V = \{(v_1, \dots, v_N) | v_n \in B, n = 1, \dots, N\} \quad (2\text{-}2)$$

式中新的符号表 B 共含 L 个符号，$B=\{b_l|l=1,\ldots,L\}$。它总共可以编出 L 个不同的码字。类似地，记$||V||=L$。为了使信源的每个输出符号序列都能分配到一个独特的码字与之对应，至少应满足关系$||V||=L\geq||U||=K$，或者$N/M\geq log_2K/log_2L$。

假若编码符号表 B 的符号数 L 与信源符号表 A 的符号数 K 相等，则编码后的码字序列的长度 N 必须大于或等于信源输出符号序列的长度 M；反之，若有$N=M$，则必须有$L\geq K$。只有满足这些条件，才能保证无差错地还原出原来的信源输出符号序列(称为码字的唯一可译性)。

但是，在这些条件下，码字序列的每个码元所载荷的平均信息量不但不能高于，反而会低于信源输出序列的每个符号所载荷的平均信息量，这与编码的基本目标是直接相矛盾的。下面的几个编码定理，提供了解决这个矛盾的方法。它们既能改善信息载荷效率，又能保证码字唯一可译。

（1） 离散无记忆信源的定长编码定理

对于任意给定的$\epsilon>0$，只要满足条件：

$$\frac{N}{M}\geq\{H(U)+\epsilon\}log_2L \qquad (2\text{-}3)$$

那么，当 M 足够大时，上述编码几乎没有失真；反之，若这个条件不满足，就不可能实现无失真的编码。式中$H(U)$是信源输出序列的符号熵。通常，信源的符号$H(U)<log_2K$，因此上述条件还可以表示为：

$$\{H(U)+\epsilon\}/log_2L\leq N/M\leq log_2K/log_2L \qquad (2\text{-}4)$$

特别，若有 $K=L$那么，只要 $H(U)<log_2K$，就可能有$N<M$，从而提高信息载荷的效率。由上面这个条件可以看出，$H(U)$离 log_2K 越远通过编码所能获得的效率改善就越显著。实质上定长编码方法提高信息载荷能力的关键是利用了渐近等分性，通过选择足够大的 M，把本来各个符号概率不等（因而 $H(U)<log_2K$）的信源输出符号序列变换为概率均匀的典型序列，而码字的唯一可译性则由码字的定长性来解决。

信息率$R=(N/M)\ log_2L$，离散无记忆信源的定长编码定理可以理解为只要 R≥H(U)，则可以实现无失真编码。反之亦然。

（2） 离散无记忆信源的变长编码定理

变长编码是指 V 的各个码字的长度不相等。只要 V 中各个码字的长度 $N_i(i=1,\ldots,\left|\left|V\right|\right|)$满足克拉夫特不等式，这$||V||$个码字就能唯一地正确划分和

译码。离散无记忆信源的变长编码定理指出，若离散无记忆信源的输出符号序列：$U\{(u_1,\dots,u_M)|u_m \in A, m=1,\dots,M\}$，式中 $A=\{a_k|k=1,\dots,K\}$，符号熵为$H(U)$，对 U 进行唯一可译的变长编码，编码字母表 B 的符号数为 L 即$B=\{b_l|l=1,\dots,L\}$,那么必定存在一种编码方法，使编出的码字 $V_i=(v_i1,\dots,v_iN_i)$，(i = 1,…，||V||)，具有平均长度α:

$$MH(U)/log_2L \le \alpha < MH(U)/log_2L+1 \qquad (2\text{-}5)$$

若 L=K，则当 $H(U)<log_2K=log_2L$ 时,必然存在编码方法，使得$\alpha<M$，满足$MH(U)\le \alpha log_2L$; $H(U)$离log_2K 越远，则α越小于 M。

以上几个编码定理，在有记忆信源或连续信源的情形也有相应的类似结果。在实际工程应用中，往往并不追求无差错的信源编码和译码，而是事先规定一个译码差错率的容许值，只要实际的译码差错率不超过这个容许值即认为满意。

具体实现唯一可译变长编码的方法很多，但比较经典的方法还是仙农编码法、费诺编码法和霍夫曼编码法。其他方法都是这些经典方法的变形和发展。所有这些经典编码方法，都是通过以短码来表示常出现的符号这个原则来实现概率的均匀化，从而得到高的信息载荷效率；同时，通过遵守克拉夫特不等式关系来实现码字的唯一可译。

其中霍夫曼编码方法的具体过程是：首先把信源的各个输出符号序列按概率递降的顺序排列起来，求其中概率最小的两个序列的概率之和，并把这个概率之和看作是一个符号序列的概率，再与其他序列依概率递降顺序排列(参与求概率之和的这两个序列不再出现在新的排列之中)，然后，对参与概率求和的两个符号序列分别赋予二进制数字 0 和 1。继续这样的操作,直到剩下一个以 1 为概率的符号序列。最后，按照与编码过程相反的顺序读出各个符号序列所对应的二进制数字组，就可分别得到各该符号序列的码字。例如，已知某个离散无记忆信源的输出符号序列及其对应的概率分布，对这些输出符号进行霍夫曼编码的具体步骤和结果如图 2.1。

由图中可以看出在码字序列中码元 0 和 1 的概率分别为 11/20 和 9/20 二者近乎相等实现了概率的均匀化。同时，由于码字序列长度满足克拉夫特不等式，因而码字是唯一可译的，不会在长的码字序列中出现划错码字的情况。

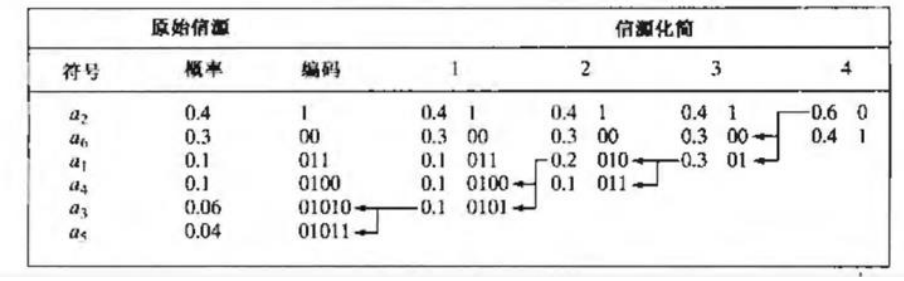

图 2.1 霍夫曼编码图示步骤

离散信源编码有香农编码、费诺编码、赫夫曼编码、游程编码、冗余位编码；连续信源编码有最佳标量量化、矢量量化；相关信源编码的预测编码、差值编码；变换编码的子带编码、小波变换。

3 算术编码的理论基础

3.1 算术编码的基本原理

算术编码是一种无失真的编码方法，能有效地压缩信源冗余度，使编成的码率趋于信的熵，它是无损压缩编码的一种。

算术编码的基本原理是：根据信源可能发现的不同符号序列的概率，把 [0，1)区间划分为互不重叠的子区间，子区间的宽度恰好是各符号序列概率。这样信源发出的不同符号序列将与各子区间一一对应， 因此每个子区间内的任意个实数都可以用来表示对应的符号序列，这个数就是该符号序列所对应的码字。显然，串符号序列发生的概率越大，对应的子区间就越宽，要表达它所用的比特数就减少，因而相应的码字就越短。

算术编码可以是静态的或者自适应的。在静态算术编码中，信源符号的概率是固定的；而在自适应算术编码中，信源符号的概率会随着信源输出的符号而改变。

具体来说，自适应算术编码在对符号序列进行扫描的过程中，可一次完成两个过程，即根据恰当的概率估计模型和当前符号序列中各符号出现的频率，自适应地调整各符号的概率估计值，同时完成编码。信源符号的概率根据编码时符号出现的频繁程度动态地进行修改，在编码期间估算信源符号概率的过程叫做建模。需要开发动态算术编码的原因是因为事先知道精确的信源概率是很难的，而且是不切实际的。当压缩消息时，我们不能期待一个算术编码器获得最大的效率，所能做的最有效的方法是在编码过程中估算概率。尽管从编码效率上看不如已知概率表的情况，但正是由于算术编码自适应的调整对个符号概率的估计值，这点比

哈弗曼编码相比，具有实时性好、灵活性高、适应性强等特点，在图像压缩、视频图像编码等领域都得到了广泛的应用。

自适应算术编码的阶数是指编码时考虑的上下文的长度。例如，0 阶自适应算术编码只考虑当前符号的概率分布，而不考虑前面出现的符号；1 阶自适应算术编码则考虑当前符号和前一个符号的联合概率分布，以此类推。一般来说，阶数越高，压缩效率越高，但也需要更多的存储空间和计算时间

本课程设计中对静态算术编码算法与 0 阶自适应算术编码算法进行 Matlab 仿真。

3.2 算术编码的特点

算术编码的优点：

(1)不必预先定义概率模型，自适应模式具有独特的优点;

(2)信源符号概率接近时，建议使用算术编码，这种情况下其效率高于霍夫曼编码;

(3)算术编码绕过了用一个特定的代码替代一个输入符号的想法，用一个浮点输出数值代替一个流的输入符号，较长的复杂的消息输出的数值中就需要更多的位数;

(4)算术编码实现方法复杂一些， JPEG 成员对多幅图像的测试结果表明，算术编码比霍夫曼编码提高了 10%左右的效率，因此在 JPEG 扩展系统中用算术编码取代霍夫曼编码。

算术编码虽然具有其独特的优点，但我们仍需要注意下面几个问题：

(1)由于实际的计算机的精度不可能无限长，运算中出现溢出是一个明显的问题，但多数机器都有 16 位、32 位或者 64 位的精度，因此这个问题可使用比例缩放方法解决。

(2)算术编码器对整个消息只产生一个码字，这个码字是在间隔[0,1)中的一个实数，因此译码器在接受到表示这个实数的所有位之前不能进行译码。

(3)算术编码也是一种对错误很敏感的编码方法，如果有一位发生错误就会导致整个消息译错。算术编码随着序列长度的增加，相应子区间的宽度也不断缩小，要表示这段子区间所需精度，直观地说就是比特数也不断增加。这不但要占用相当大的存储空间，还增加了编码延时，这对实时系统是十分不利的。为了解

决这些难点，针对不同的应用方向，人们对传统的算术编码方法进行了改进，在保证足够精度的前提下，提高了编码速度。基于算术编码算法人们提出了二进制自适应的算术编码以及 MO 算术编码器分别在软件及上提高编码的效率。

3.3 算术编码的分析过程

在算术编码中，消息用 0 到 1 之间的实数进行编码，算术编码用到两个基本的参数:符号的概率和它的编码间隔。信源符号的概率决定压缩编码的效率，也决定编码过程中信源符号的间隔，而这些间隔包含在 0 到 1 之间。编码过程中的间隔决定了符号压缩后的输出。算术编码的过程，实际上就是依据信源符号的发生概率对码区间分割的过程。

静态算术编码和自适应型算术编码在编码前都需要初始化概率空间，静态算术编码的字符概率是固定的，因此找到相应的概率空间可直接按区间分割进行编码；自适应型算术编码在编码前需要统计输入的文本信息的符号类型和每个符号的个数，期初假定每个符号概率相等，然后输入一个符号后，找到相应的概率空间所有的符号概率会进行更新，然后依次规律对输入信息进行编码，算数编码的分析框图如下：

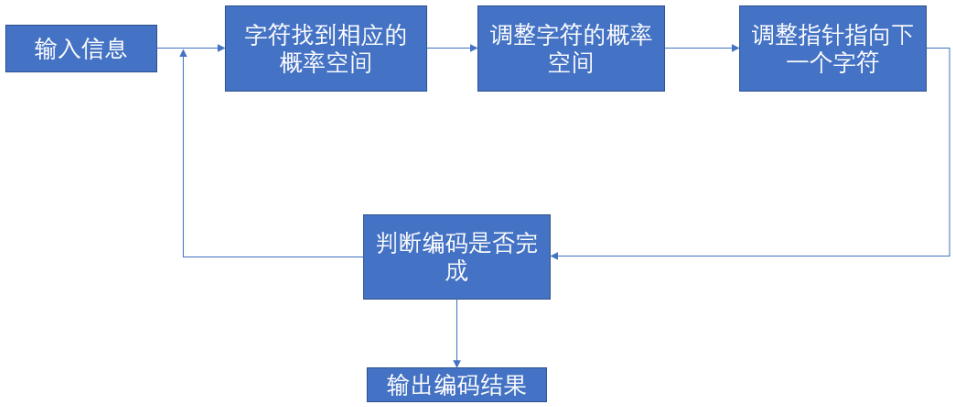

图 3.1 算数编码的分析框图

解码过程，读取编码结果，找到所属区间范围从而译出码字。静态型算术编码的编码值是变化的然后找所对应的区间;自适应型算术编码的编码值是不变的，只需改变概率区间，然后用此编码值找到所对应的区间，从而译出码字。算数解码的分析框图如下：

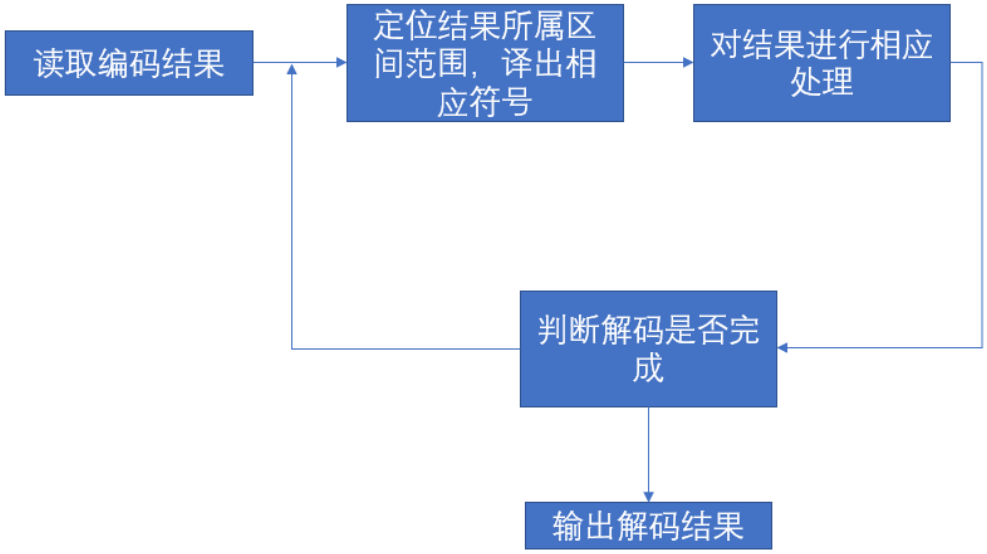

图 3.2 算数解码的分析框图

3.4 算术编码的举例

3.4.1 静态算术编码

算术编码将被编码的信源符号流表示成实数半开区间[0,1)中的一个数值间隔，这个间隔随着信息流中每一个信源符号的加入逐步减小,每次减小的程度取决于当前加入的信源符号的概率。概率高者减少的程度低，概率低者减少的程度高。符号流越长，代表它的间隔越小，编码表示这一间隔所需的位数就越多。从算术编码过程中产生的数值可以被唯一地解码，精确地恢复原始的信源符号流。

3.4.2 静态算术编码模型

设信源符号集由 3 个信源符号{a1，a2，a3}组成，根据已知的信源符号概率在[0.1)区间上为它们分配相应的子区间，子区间大小与概率成正比。其概率分布及对应的子区间如表 3.1 所示：

表 3.1 信源符号概率分布及对应的子区间

信源符号	概率	子区间范围
a_1	0.4	[0.0 , 0.4)
a_2	0.4	[0.4 , 0.8)
a_3	0.2	[0.8 , 1.0)

$$range = high - low$$
$$high = low + range \times high\_range \quad (3\text{-}1)$$
$$low = low + range \times low\_range$$

用 $range$ 表示编码输出数值落入的间隔，$high$ 为$range$ 的上界，low 为下界,初始时 $high$ 为 1，low为 0。用$high_range$ 表示某符号子区间的上界，low_range 为下界。每一信源符号被编码后,根据公式(3-1)计算$high$、low 及 $range$ 的新值，并根据概率分布重新划分被编码符号的子区间。最后一个符号得到的间隔内的任何一个实数代表整个符号流。对符号流“a1a2a3a2a1”进行算术编码的过程如图 3.3 所示用 0.305 就可以唯一地代表这个符号流。

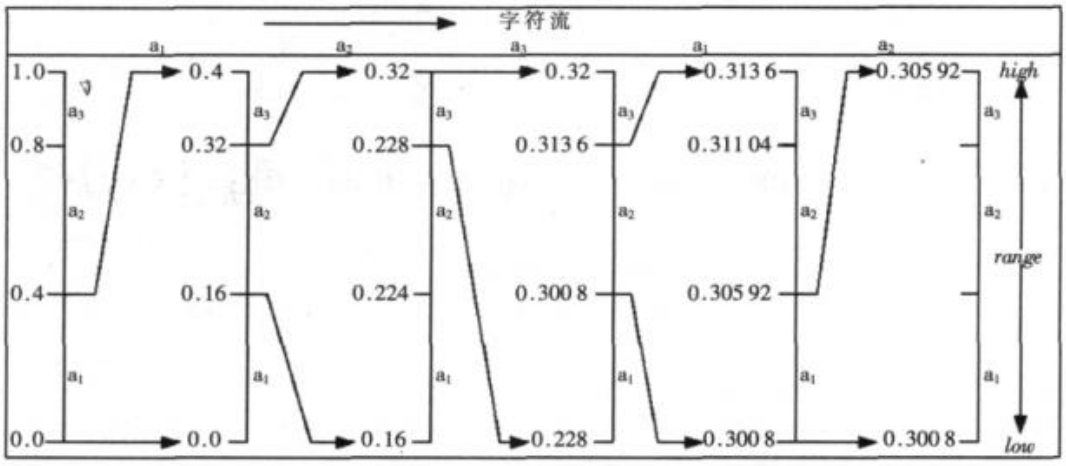

图 3.3 静态算数编码过程

3.4.3 自适应算术编码

根据编码过程中信源符号是否更新概率分布算术编码分为静态模型和自适应模型。静态模型的缺点是首先需要在编码前对信源符号的分布进行统计，会消耗大量时间：其次符号的概率是该符号在整个信息中的概率，根据信息熵原理，不利于对信息的压缩。而自适应模型随着符号的读入，实时动态更新符号的概率分布，能统计出某个符号在局部的出现概率或某个符号相对于某一上下文的出现概率，压缩效果好于静态模型。自适应编码中，初始时假设各个符号的概率相同并平均分配区间[0,1)。随着符号的读入，相应地更新其概率值，与之对应的子区间亦随之改变。若有新的符号出现则加入符号集即可。仍设信源符号集由 3 个符号[a1,a2,a3)组成，对于符号流“a1a2a3a2a1”各个符号概率更新、子区间比例变化符号累计及间隔变化过程如表 3. 2 所示。

最后一个符号输入后不再划分子区间，得到读入 al后的区间[0.2389,0.2404)，

即编码结果的输出数值区间，如 0.24 就可以作为符号流编码结果。同样类似的解码过程可以唯一地解出原符号流。

表 3.2 自适应算数编码过程

更新信源	初始概率	子区间范围	读入 a_1	子区间范围	读入 a_2	子区间范围	读入 a_3	子区间范围	读入 a_2	子区间范围	读入 a_1
a_1	1/3	[0.0,0.333 3)	1/2	[0.0,0.5)	2/5	[0.0,0.40)	1/3	[0.0,0.333 3)	2/7	[0.0,0.285 7)	3/8
a_2	1/3	[0.333 3,0.666 7)	1/4	[0.5,0.75)	2/5	[0.40,0.8)	1/3	[0.333 3,0.666 7)	3/7	[0.285 7,0.714 3)	3/8
a_3	1/3	[0.6667,1.0)	1/4	[0.75,1.0)	1/5	[0.8,1.0)	1/3	[0.666 7,1.0)	2/7	[0.714 3,1.0)	2/8
符号累计	3		4		5		6		7		8
range	[0.0,1.0)		[0.0,0.333 3)		[0.166 7,0.25)		[0.233 4,0.25)		[0.238 9,0.244 5)		[0.238 9,0.240 4)

如果编码时考虑符号之间的相关性，把多个符号按照不同的上下文结构组合在一起，当作一个编码单元进行自适应算术编码，就可以进一步提高编码效率用“阶”表示上下文相关符号序列的长度，那么 1 阶上下文自适应统计的就是符号在某个特定的符号后面出现的概率，同样 2 阶、3 阶上下文自适应统计的是符号在某两个、三个特定符号后面出现的概率。使用多阶算术编码，使得同一符号可以在多个动态统计的上下文概率表中取得概率值较大的进行编码，从而使总熵值更小。如在 1 阶自适应编码中在已经编码的 10000 个符号后刚刚编码的符号是 a1，下一个要编码的符号是 a2，在之前的 10000 个符号中统计到出现了 100 个 al 和 a2，a2 出现在 a1 之后的次数是 10 次，那么 a2 在 a1 之后的概率是 10/100，这一概率对应的熵值是 -log2(10/100)=3.3219bit，远小于 0 阶的熵值 -log2(100/10000)=6.6439bit。采用 2 阶、3 阶或更高阶次的自适应算术编码可以获得更高的压缩率。

3.4.4 0 阶自适应算术编码模型

用 $scale$ 表示已读入符号总数，$count$表示输入 ai 的总数，读入符号 ai 时，$scale$加 1，$count$加 1。用 $high_Newrange$和 $low\_Newrange$ 表示 ai 更新后对应的子区间上下界， ai 新的概率用$count/scale$来表示，进而得到更新后的$high_range$和 low_range。$high$ 和 low 表示编码输出数值$range$ 的上下界，其初始值分别为 1 和 0，则可以通过公式(3-2)确定 $range$的范围。

$$range = high - low$$
$$high = low + range \times high\_Newrange \quad (3\text{-}2)$$
$$low = low + range \times low\_Newrange$$

4 算数编码 MATLAB 仿真实现

4.1 MATLAB 仿真程序实现

MATLAB 是由美国 mathworks 公司发布的主要面对科学计算、可视化以及交互式程序设计的高科技计算环境。它将数值分析、矩阵计算、科学数据可视化以及非线性动态系统的建模和仿真等诸多强大功能集成在一个易于使用的视窗环境中，为科学研究、工程设计以及必须进行有效数值计算的众多科学领域提供了一种全面的解决方案，并在很大程度上摆脱了传统非交互式程序设计语言(如 C、Fortran) 的编辑模式，代表了当今国际科学计算软件的先进水平。

MATLAB 由一系列工具组成。这些工具方便用户使用 MATLAB 的函数和文件，其中许多工具采用的是图形用户界面。包括 MATLAB 桌面和命令窗口、历史命令窗口、编辑器和调试器、路径搜索和用于用户浏览帮助、工作空间 、文件的浏览器。随着 MATLAB 的商业化以及软件本身的不断升级，MATLAB 的用户界面也越来越精致，更加接近 Windows 的标准界面人机交互性更强，操作更简单。而且新版本的 MATLAB 提供了完整的联机查询、帮助系统极大的方便了用户的使用。简单的编程环境提供了比较完备的调试系统，程序不必经过编译就可以直接运行，而且能够及时地报告出现的错误及进行出错原因分析。

MATLAB71 版本是在 2005 年更新的，本文主要应用 MATLAB71 中发布的影像处理工具箱中的相关函数和命令来实现基于算术编码实现图像压缩理论算法的仿真。MATLAB7.1 是套功能十分强大的工程计算及数据分析应用软件,广泛应用于工业、电子、控制、信号及图像处理等各领域。MATLAB7.1 本身除了提供强大的图形绘制和输出功能外，同时还发布了影像处理工具箱(Image Processing Toolbox)专门用于图像的处理在通常情况下，可以用它来代替底层编程语言，如 C 和 C++。在计算要求相同的情况下，使用 MATLAB 的编程工作量会大大减少。

4.2 仿真设计流程图

4.2.1 静态算数编码仿真设计

编码：静态算术编码输入的字符类型固定，每个字符的概率也是固定的。输入的自符类型有“abcdef”每次输入字符，更新字符的起始、终止区间。待最后一个字符编码完成后，取起始值和终止值的中点作为编码的结果输出。

信道中传输：将输出的结果转化为 2 进制数在信道中传输。信道中传输的二进制数，转化为十进制数后进行译码。

解码：译码的时候，读取编码的输出结果，找到所在的区间，依次译出编码前输入的字符信息。

静态算数编码仿真设计流程图如图 4.1。

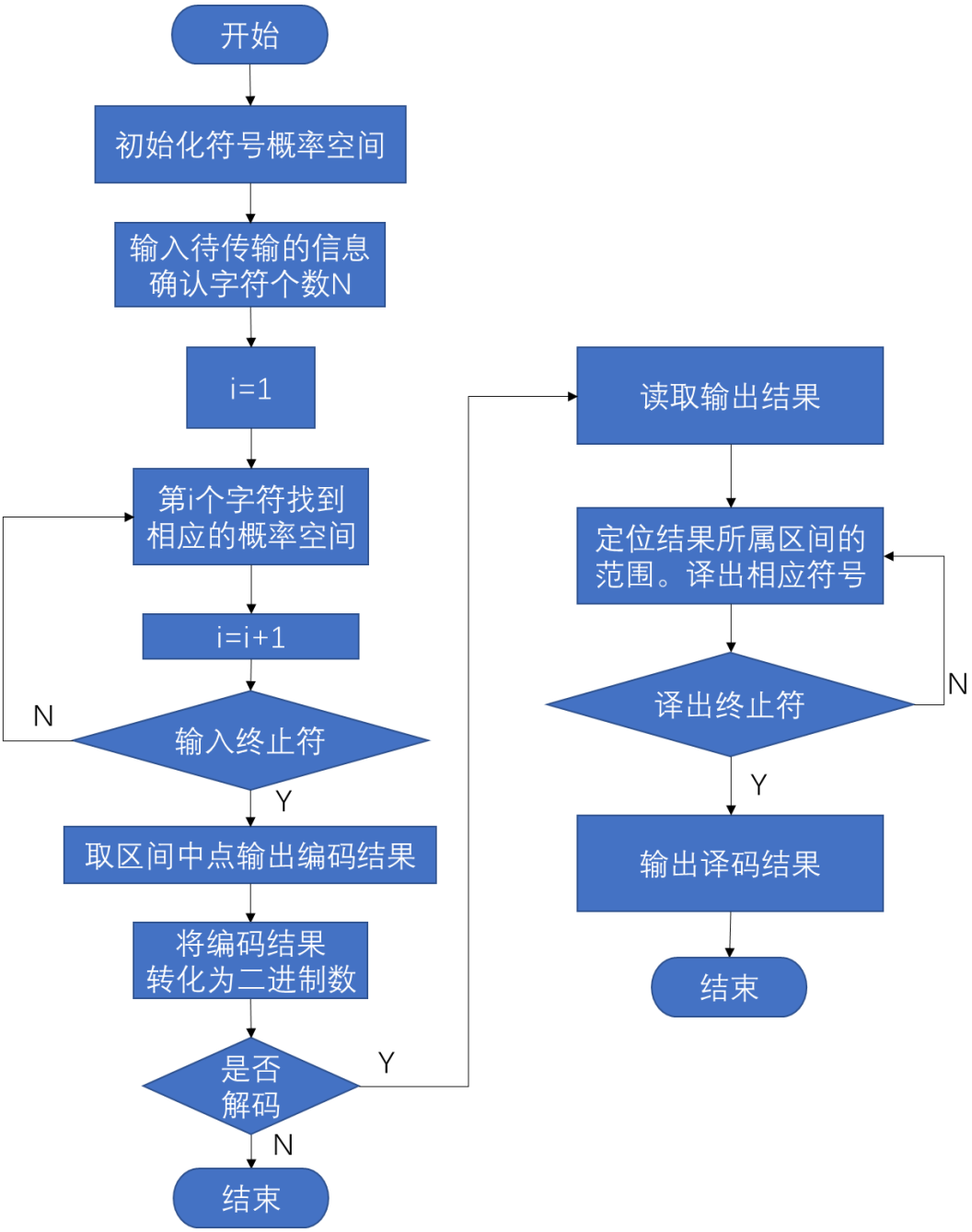

图 4.1 静态算数编码仿真设计流程图

4.2.2 0 阶自适应算数编码仿真设计

编码：自适应算术编码的概率模型不固定，先统计编码的符号类型，以及每个符号的个数。编码前，假设每个符号的概率是相等的，然后每次输入一个字符，相应的字符概率发生变化，直至编出最后一个码字，选取区间中间结果作为编码的输出。

信道中传输：将输出的结果转化为 2 进制数在信道中传输。信道中传输的二进制数，转化为十进制数后进行译码。

解码：译码时，读取中间结果，找到所属概率区间，译出码字，然后变更概率区间，重新定位码字。直至译出最后一个码字。

0 阶自适应算数编码仿真设计流程图如图 4.2。

无论是静态型还是自适应型算术编码在编码前都要初始化概率空间，但二者在编码时的概率空间却不一样，前者固定不变，后者概率随输入字符变化而变化。然后进行区间分割，将字符串编成一个浮点小数，然后转化为二进制，作为结果之后选择是否译码，确定是否译出信源符号。

4.3 通信系统设计与仿真细节描述

4.3.1 十进制小数转二进制小数

算数编码得到的十进制小数转换成二进制小数采用“乘 2 取整，顺序排列”法。具体做法是：用 2 乘十进制小数，可以得到积，将积的整数部分取出，再用 2 乘余下的小数部分，又得到一个积，再将积的整数部分取出，如此循环往复，直到积中的小数部分为零。但有时积中的小数在取出有限位数后，仍然不能为零，这就涉及到十进制数转二进制数的精度问题。

精度问题会影响编码结果以及译码结果，较低的精度会导致译码错误，过高的精度又会浪费资源，因此实际应用中需要选择合适的精度。在这次通信系统设计与仿真中，我选取了 40 位码元作为 2 进制数的位数，在信道中传输。

4.3.2 终止符

在解码过程中，接收方并不知道所要解码的符号的个数，所以需要预设一个终止符，当接收方解码出终止符时，即刻停止解码。因此在算数编码中，需要预留概率段作为终止符概率。

4.3.2 概率初始化

静态算数编码统计各类符号出现的频率作为符号的概率。0 阶自适应算数编码假定初始符号的概率相同，概率随着输入的符号而改变。

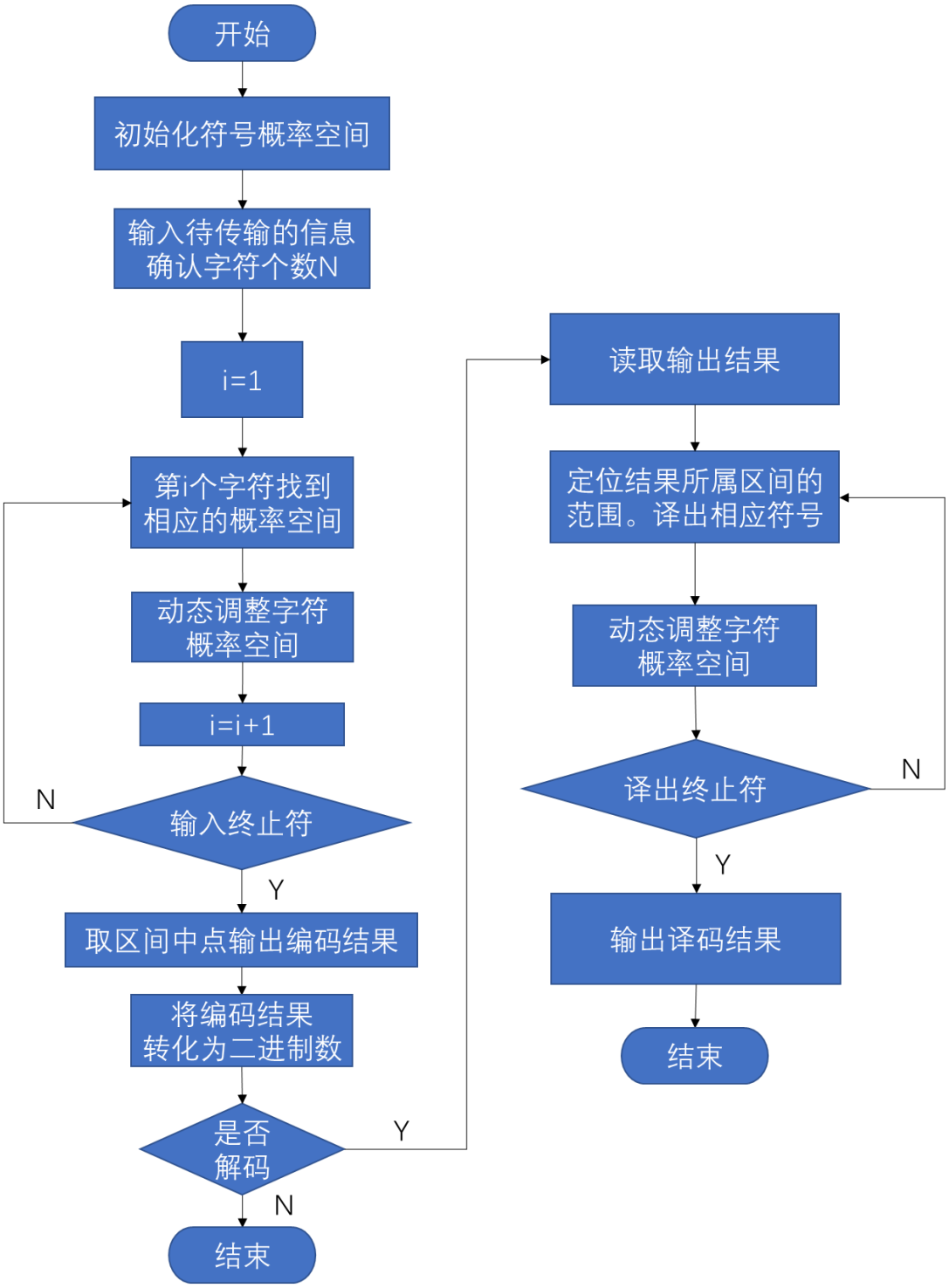

图 4.2 0 阶自适应算数编码仿真设计流程图

4.3 通信系统仿真效果图

假设要传输的数据为“abcdeedcbaf”。

4.3.1 静态算数编码通信系统仿真效果图

图 4.3 展示了静态算数编码通信系统仿真效果图，流程为输入要压缩的字符串，取编码区间的中点作为编码结果，将十进制数转化为二进制码，接受端将二进制码还原为二进制数，最终得到译码结果。

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    '可以输入的字符仅限于：a,b,c,d,e,f,其中f作为终止符'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
算数编码前的字符串：
abcdeedcbaf
编码区间的起始值：
   0.049318984077539
编码区间的终止值：
   0.049318987666597
编码区间中点作为编码结果：
   0.049318985872068
编码结果的二进制码向量：
  1 至 20 列
     0     0     0     0     1     1     0     0     1     0     1     0     0     0     0     0     0     0     1     0
  21 至 40 列
     1     0     1     1     0     1     0     0     0     1     1     1     0     1     1     0     0     1     0     0
二进制码向量还原为十进制：
   0.049318985871651
算数解码后的字符串：
abcdeedcbaf
```

图 4.3 静态算数编码通信系统仿真效果图

4.3.2 0 阶自适应算数编码通信系统仿真效果图

图 4.4 展示了 0 阶自适应算数编码通信系统仿真效果图，流程同静态算数编码，在此不再赘述。

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    '可以输入的字符仅限于：a,b,c,d,e,f,其中f作为终止符'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
算数编码前的字符串：
abcdeedcbaf
编码区间的起始值：
   0.061799876159995
编码区间的终止值：
   0.061799876343527
编码区间中点作为编码结果：
   0.061799876251761
编码结果的二进制码向量：
  1 至 20 列
     0     0     0     0     1     1     1     1     1     1     0     1     0     0     1     0     0     0     0     1
  21 至 40 列
     1     1     0     1     1     1     0     1     1     1     1     1     0     1     1     0     0     1     0     1
二进制码向量还原为十进制：
   0.061799876250916
算数解码后的字符串：
abcdeedcbaf
```

图 4.4 0 阶自适应算数编码通信系统仿真效果图

可以看到二进制码向量的精度影响最终解码后的十进制小数。同时，算数编码前后的字符串一致，验证了算术编码是一种无失真编码。

5 性能分析

5.1 算法的仿真性能分析

在本节中主要对静态算数编码和 0 阶自适应算数编码进行仿真，假设信道为理想信道，误码率为 0，使用相同的要传输的数据，对比两种不同的编码方式对数据压缩的影响，如表 5.1 所示，其中二进制压缩码只展示前 12 位码。

表 5.1 两种算术编码算法对比

输入符号	静态算数编码		0 阶自适应算数编码	
	十进制压缩码	二进制压缩码	十进制压缩码	二进制压缩码
a	0.500000000000000	100000000000	0.083333333333333	000101010101
ab	0.375000000000000	011000000000	0.059523809523810	000011110011
abc	0.203703703703704	001101000010	0.061011904761905	000011111001
abcd	0.107421875000000	000110111000	0.061673280423280	000011111100
abcde	0.062240000000000	000011111110	0.061789021164021	000011111101
abcdef	0.039984139231824	000010100011	0.061804052429052	000011111101
abbbbb	0.133176868998628	001000100001	0.055615680615681	000011100011
eeeeee	0.500000000000000	100000000000	0.799350649350649	110011001010
abbcdefffff	0.010699581438301	000000101011	0.057401394213150	000011101011
abccbaabc	0.214271198496164	001101101101	0.060981627499485	000011111001
abcccdcab	0.089504207920196	000101101110	0.061261688047402	000011111010
aaabbbccc	0.019178986943047	000001001110	0.008953100470958	000000100100
bcdbcdbcd	0.192323324696439	001100010011	0.253660960952628	010000001110
fcdabcfcaf	0.844457486200000	110110000010	0.892940653922797	111001001001
cbacbdddd	0.360254836702764	010111000011	0.358703664060807	010110111101
ecfbacdefe	0.609235469600000	100110111111	0.735465688412117	101111000100

可以看到，实际情况中，不同符号的概率分布无法事先确定，而自适应算数编码可以动态的调整符号概率，使得模型的符号概率更加符合真实值，以至于有更好的压缩效果。

6 总结

1.首先介绍了信源编码的理论基础，引出离散无记忆信源的变长编码可以压缩源码的理论依据。

2.通过仿真发现在算术编码中，编码结果的精度，即十进制小数的有效位数和二进制码的个数都会很大程度上影响通信系统接收端解码的结果，十进制小数的有效位数越多，二进制码的个数越多，解码出的十进制小数也越精确，然而过高的精度会一定程度上占用计算机的性能。因此也印证了早期算术编码思想出现时，由于硬件配置无法满足，导致之后很长时间才得以进行大规模的应用。

3.自适应算数编码相对于静态算数编码的优点在于实际情况中，不同符号的概率分布无法事先确定，而自适应算数编码可以动态的调整符号概率，使得模型的符号概率更加符合真实值，以至于有更好的压缩效果。但自适应算数编码对计算机的性能要求更高，随着自适应算数编码的阶数增加，系统内存占用增多。本课程设计中，我只对静态算数编码与 0 阶自适应算数编码进行了仿真实现，未来可以在我目前研究的基础上，进一步对更高阶数的自适应算术编码进行研究。

7 心得体会

这学期的 MATLAB 通信系统仿真课程让我学到了很多，从数据的各种运算方式，符号的运算到矩阵的分析和处理，程序的设计，绘图，在老师的课程带领下迅速掌握了 MATLAB 的使用方法。在理论课程的学习中，使我在短时间内可以熟练的上手操作，并且激发了我对于 MATLAB 学习的欲望。

论文开题前，涉及到论文的选题。由于在四月份的时候，深感复习效率之低，因此我联系了梦校的老师，并且得到了积极的面试回复，给予我考核任务去做。也由此开始了我长达两个月的任务考核。而在所做任务中，有一项是基于 Transformer 的压缩编码的模型代码复现工作，所用到的底层算法就是本次课程设计实现的算术编码，由于原始论文中的代码更多的是对封装好的库的调用，因此我便有了实现算数编码的想法。由此，本次课程设计应运而生。

在这个课程设计中，我学到了很多算术编码算法的知识，也学习了很多 MATLAB 的使用技巧，认识到了信源编码的基本目的是减少信源输出符号序列的剩余度、提高码字序列中码元的平均信息量等等相关知识。

在编程的过程中，又一次熟悉了通信系统仿真软件 MATLAB 的基本使用方法，能够在该软件的使用过程中解决随时出现的问题，最重要的是提高了我综合运用所学知识的能力和计算机的编程能力，为今后的工作和学习积累了宝贵的经验，对于以后的毕业设计以及之后的工作都有相当重要的意义。

从刚开始一拿到这个论文时的迷茫，不知所措，再到确定选题，开始每天研究代码的编写，不断的去学习新的知识，再到代码终于可以运行出结果。我能清晰的感觉出自己在这个过程中的成长，通过这一次结课论文的学习，我相信自己一定可以在以后的学习中变得更加严谨、认真，这将对我以后的科研之路至关重要。

参考文献

[1] 刘福来.MATLAB 与无线电信号处理分析[M].西安：西安电子科技大学出版社,2020.

[2] 范昌信.通信原理（第七版）[M].北京：国防工业出版社,2013.

[3] 冯桂,林其伟,陈东华.信息论与编码技术（第二版）[M].北京：清华大学出版社,2011.

[4] 谢林,虞露,仇佩亮.基于上下文的自适应二进制算术编码研究[J].浙江大学学报(工学版),2005(06):910-914.

[5] 罗钧,张国彬.JPEG2000 中的二进制算术编码及其 DSP 实现[J].重庆大学学报(自然科学版),2003(04):34-37.

[6] 张杰,童胜.H.264/AVC 中二进制算术编码的分析与研究[J].电子科技，2003(24):28-31.

附　　录

基于静态算数编码的通信系统设计仿真的部分源程序代码

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%此代码实现静态算数编码
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
menu={...'输入字符串的要求限制：'
'可以输入的字符仅限于：a,b,c,d,e,f '};
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
disp(menu)
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
string_s='ecfbacdefe';%输入的字符串
disp('算数编码前的字符串：')
disp(string_s)
%统计输入字符串的概率：
pa=s2p(string_s,'a');pb=s2p(string_s,'b');
pc=s2p(string_s,'c');pd=s2p(string_s,'d');
pe=s2p(string_s,'e');pf=s2p(string_s,'f');
k=length(string_s);
a1=0;a2=1;l=a2-a1;%计算字符串编码区间的长度
for i=1:k %开始算术编码
          if string_s(i)=='a'
              aa1=a1;aa2=a1+l*pa;
              a1=aa1;a2=aa2;l=a2-a1;
          elseif string_s(i)=='b'
              aa1=a1+l*pa;aa2=a1+l*(pa+pb);
              a1=aa1;a2=aa2;l=a2-a1;
          elseif string_s(i)=='c'
              aa1=a1+l*(pa+pb);aa2=a1+l*(pa+pb+pc);
              a1=aa1;a2=aa2;l=a2-a1;
          elseif string_s(i)=='d'
              aa1=a1+l*(pa+pb+pc);aa2=a1+l*(pa+pb+pc+pd);
              a1=aa1;a2=aa2;l=a2-a1;
          elseif string_s(i)=='e'
              aa1=a1+l*(pa+pb+pc+pd);aa2=a1+l*(pa+pb+pc+pd+pe);
              a1=aa1;a2=aa2;l=a2-a1;
          elseif string_s(i)=='f'
              aa1=a1+l*(pa+pb+pc+pd+pe);aa2=a1+l*(pa+pb+pc+pd+pe+pf);
              a1=aa1;a2=aa2;l=a2-a1;
          end
end
disp('编码区间的起始值：')
disp(a1)
disp('编码区间的终止值：')
```

```
disp(a2)
disp('编码区间中点作为编码结果：')
disp((a1+a2)/2)
disp('编码结果的二进制码向量：')
y=d2b((a1+a2)/2,40);
disp(y)
disp('二进制码向量还原为十进制：')
x=b2d(y,40);
disp(x)
decoder = blanks(k); % A string of k spaces
a1=0;a2=1;l=a2-a1;
for i=1:k %开始算术解码
      if x>=a1&&x<(a1+l*pa)
          decoder(i)='a';
          aa1=a1;aa2=a1+l*pa;
          a1=aa1;a2=aa2;l=a2-a1;
      elseif x>=(a1+l*pa)&&x<(a1+l*(pa+pb))
          decoder(i)='b';
          aa1=a1+l*pa;aa2=a1+l*(pa+pb);
          a1=aa1;a2=aa2;l=a2-a1;
      elseif x>=(a1+l*(pa+pb))&&x<(a1+l*(pa+pb+pc))
          decoder(i)='c';
          aa1=a1+l*(pa+pb);aa2=a1+l*(pa+pb+pc);
          a1=aa1;a2=aa2;l=a2-a1;
      elseif x>=(a1+l*(pa+pb+pc))&&x<(a1+l*(pa+pb+pc+pd))
          decoder(i)='d';
          aa1=a1+l*(pa+pb+pc);aa2=a1+l*(pa+pb+pc+pd);
          a1=aa1;a2=aa2;l=a2-a1;
      elseif x>=a1+l*(pa+pb+pc+pd)&&x<(a1+l*(pa+pb+pc+pd+pe))
          decoder(i)='e';
          aa1=a1+l*(pa+pb+pc+pd);aa2=a1+l*(pa+pb+pc+pd+pe);
          a1=aa1;a2=aa2;l=a2-a1;
      elseif x>=a1+l*(pa+pb+pc+pd+pe)&&x<(a1+l*(pa+pb+pc+pd+pe+pf))
          decoder(i)='f';
          aa1=a1+l*(pa+pb+pc+pd+pe);aa2=a1+l*(pa+pb+pc+pd+pe+pf);
          a1=aa1;a2=aa2;l=a2-a1;
      end
end
disp('算数解码后的字符串：')
    disp(decoder)
```

基于自适应算数编码的通信系统设计仿真的部分源程序代码

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%此代码实现自适应算数编码
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
menu={...'输入字符串的要求限制：'
'可以输入的字符仅限于：a,b,c,d,e,f '};
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
disp(menu)
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
string_s='ecfbacdefe';%输入的字符串
disp('算数编码前的字符串：')
disp(string_s)
%假定初始各个字符等概率
string_s0='abcdef';
k=length(string_s);
a1=0;a2=1;l=a2-a1;%计算字符串编码区间的长度
for i=1:k %开始算术编码
          if string_s(i)=='a'
              %统计已经输入字符的概率：
              pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
              pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
              pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
              aa1=a1;aa2=a1+l*pa;
              a1=aa1;a2=aa2;l=a2-a1;
              %统计已经输入字符：
              string_s0 = strcat(string_s0,'a');
          elseif string_s(i)=='b'
              pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
              pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
              pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
              aa1=a1+l*pa;aa2=a1+l*(pa+pb);
              a1=aa1;a2=aa2;l=a2-a1;
              string_s0 = strcat(string_s0,'b');
          elseif string_s(i)=='c'
              pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
              pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
              pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
              aa1=a1+l*(pa+pb);aa2=a1+l*(pa+pb+pc);
              a1=aa1;a2=aa2;l=a2-a1;
              string_s0 = strcat(string_s0,'c');
          elseif string_s(i)=='d'
              pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
              pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
              pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
```

```
            aa1=a1+l*(pa+pb+pc);aa2=a1+l*(pa+pb+pc+pd);
            a1=aa1;a2=aa2;l=a2-a1;
            string_s0 = strcat(string_s0,'d');
        elseif string_s(i)=='e'
            pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
            pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
            pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
            aa1=a1+l*(pa+pb+pc+pd);aa2=a1+l*(pa+pb+pc+pd+pe);
            a1=aa1;a2=aa2;l=a2-a1;
            string_s0 = strcat(string_s0,'e');
        elseif string_s(i)=='f'
            pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
            pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
            pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
            aa1=a1+l*(pa+pb+pc+pd+pe);aa2=a1+l*(pa+pb+pc+pd+pe+pf);
            a1=aa1;a2=aa2;l=a2-a1;
            string_s0 = strcat(string_s0,'f');
        end
end
disp('编码区间的起始值：')
disp(a1)
disp('编码区间的终止值：')
disp(a2)
disp('编码区间中点作为编码结果：')
disp((a1+a2)/2)
disp('编码结果的二进制码向量：')
y=d2b((a1+a2)/2,40);
disp(y)
disp('二进制码向量还原为十进制：')
x=b2d(y,40);
disp(x)
decoder = blanks(k); % A string of k spaces
%假定初始各个字符等概率
string_s0='abcdef';
pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
a1=0;a2=1;l=a2-a1;
for i=1:k %开始算术解码
    if x>=a1&&x<(a1+l*pa)
        decoder(i)='a';
        aa1=a1;aa2=a1+l*pa;
        a1=aa1;a2=aa2;l=a2-a1;
        string_s0 = strcat(string_s0,'a');
```

```
        pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
        pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
        pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
    elseif x>=(a1+l*pa)&&x<(a1+l*(pa+pb))
        decoder(i)='b';
        aa1=a1+l*pa;aa2=a1+l*(pa+pb);
        a1=aa1;a2=aa2;l=a2-a1;
        string_s0 = strcat(string_s0,'b');
        pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
        pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
        pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
    elseif x>=(a1+l*(pa+pb))&&x<(a1+l*(pa+pb+pc))
        decoder(i)='c';
        aa1=a1+l*(pa+pb);aa2=a1+l*(pa+pb+pc);
        a1=aa1;a2=aa2;l=a2-a1;
        string_s0 = strcat(string_s0,'c');
        pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
        pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
        pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
    elseif x>=(a1+l*(pa+pb+pc))&&x<(a1+l*(pa+pb+pc+pd))
        decoder(i)='d';
        aa1=a1+l*(pa+pb+pc);aa2=a1+l*(pa+pb+pc+pd);
        a1=aa1;a2=aa2;l=a2-a1;
        string_s0 = strcat(string_s0,'d');
        pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
        pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
        pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
    elseif x>=a1+l*(pa+pb+pc+pd)&&x<(a1+l*(pa+pb+pc+pd+pe))
        decoder(i)='e';
        aa1=a1+l*(pa+pb+pc+pd);aa2=a1+l*(pa+pb+pc+pd+pe);
        a1=aa1;a2=aa2;l=a2-a1;
        string_s0 = strcat(string_s0,'e');
        pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
        pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
        pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
    elseif x>=a1+l*(pa+pb+pc+pd+pe)&&x<(a1+l*(pa+pb+pc+pd+pe+pf))
        decoder(i)='f';
        aa1=a1+l*(pa+pb+pc+pd+pe);aa2=a1+l*(pa+pb+pc+pd+pe+pf);
        a1=aa1;a2=aa2;l=a2-a1;
        string_s0 = strcat(string_s0,'f');
        pa=s2p(string_s0,'a');pb=s2p(string_s0,'b');
        pc=s2p(string_s0,'c');pd=s2p(string_s0,'d');
        pe=s2p(string_s0,'e');pf=s2p(string_s0,'f');
    end
```

```
end
disp('算数解码后的字符串：')
    disp(decoder)
```

其他函数实现

```
function binStr = d2b(decNum, binWidth)
% 这个函数将一个十进制小数转换为一个指定位数的二进制数
% 输入: decNum - 一个十进制小数
%       binWidth - 一个正整数，表示二进制数的位数
% 输出: binStr - 一个字符向量，表示二进制数
binStr = [];
for i =1:binWidth
    decNum=decNum*2;
    if decNum>=1
      binStr(i)=1;
      decNum=decNum-1;
    elseif decNum<1
       binStr(i)=0;
    end
end


function decNum = b2d(binStr, binWidth)
% 这个函数将一个二进制数转换为一个十进制小数
% 输入: binStr - 一个二进制数
%          binWidth - 一个正整数，表示二进制数的位数
% 输出: decNum - 一个十进制数
decNum=0;
for i =1:binWidth
    decNum=decNum+binStr(i)*2^(-i);
end


function p = s2p(Str, zimu)
% 输入: Str - 输入字符串
%          zimu - 要统计概率的字母
% 输出: p - 该字母的概率
wigth=length(Str);
q=0;
for i =1:wigth
```

```
    if Str(i)==zimu
       q=q+1;
    end
end
p=q/wigth;
```

小组成员列表

组长：202012544　姓名：韩泽华

学号	姓名	负责内容	贡献率
202012544	韩泽华	全部	100%