第 34 卷 第 4 期 应 用 数 学 学 报 Vol. 34 No. 4
2011 年 7 月 ACTA MATHEMATICAE APPLICATAE SINICA July, 2011

循环码译码的Dixon结式方法 *

李耀辉 赵海豹 马春芽

(天津工程师范学院计算机科学系，天津 300222)

(E-mail: phillipplee@163.com)

摘 要 针对纠错码译码就是非线性方程组的求解问题，提出利用 Dixon 结式方法对译码方程进行消元以得到接收数据中的错位多项式．首先，根据纠错码的纠错能力和接收数据得到伴随式矩阵并通过该矩阵的秩确定接收码字中错误位的个数．然后，根据错位个数和伴随多项式构造译码方程．译码时，将其中一个错位变元作为隐藏变元，利用 Dixon 结式方法进行消元．最后，得到的 Dixon 结式就是关于隐藏变元的多项式．该多项式去掉多余因子后就是错位多项式，利用 Chien 搜索法即可求解出错误位置．当错位较多时，采用逐次计算结式的方法以筛除计算过程中的多余因子和重因子．另外，根据不同错位个数得到的错位多项式，提出了构造一类循环码错位多项式符号解的猜想，该猜想可以大大提高译码效率．实验验证了结式理论在纠错码译码方面的应用是有效的且有助于降低对芯片性能的要求．

关键词 Dixon 结式；纠错码；译码；伴随式

MR(2000) 主题分类 04B35

中图分类 O236.2

1 引言

纠错码技术可以提高通信质量，降低信道干扰带来的误码率．在有噪声的信道上，利用纠错码可以实现信息的可靠传输．但是，纠错是有代价的，它是通过在传输的数据中增加冗余实现．如何兼顾可靠性和编码效率是纠错码编码所要考虑的问题．国内外对于纠错码理论的研究课题主要有两方面：(1) 构造性能良好的纠错码，即要求码率 k/n 和反映纠错能力的最小距离 d 愈大愈好；(2) 寻求好的译码算法，以提高译码的效率 [1,2]，其中，译码问题是纠错码理论与实际中最关键、最有吸引力的问题．信道编码的理论再好，如果译码时没有可行的高效算法也是无用的．译码步骤主要有 3 步，即：第 1 步根据接收到的数据 R 计算出译码多项式；第 2 步由伴随式找出错误图样 $\widetilde{e}(x)$; 第 3

本文 2006 年 1 月 9 日收到．2007 年 9 月 24 日收到修改稿．

* 国家 973 计划 (NKBRSF-2004CB318003)，天津工程师范学院引进人才基金 (KYQD06005) 和天津工程师范学院科研基金 (KJ20080039) 资助项目．

步，根据编码理论，由 $R(x)-\widetilde{e}(x)$ 得到最可能发送的码字 $\widetilde{C}(x)$. 译码的主要过程是第 2 步的如何求解错位多项式及错值多项式. 该问题的实质是如何根据纠错码的伴随多项式及其有限域上的相关约束条件求解关于错位和错值的非线性方程组.

对于该问题，许多学者采用非线性消元的 Gröbner 基理论进行研究 [3−7]. 利用 Gröbner 基分析纠错码编译码主要基于两方面的原因：(1) 编码过程中，所有编码多项式都可由生成多项式生成，即码多项式在生成多项式的生成理想中. 编码问题可以看作理想成员的判定和计算问题. (2) 译码问题就是设法通过消元求解出非线性代数方程组的解. 多项式系统生成理想字典序的 Gröbner 基恰具有消元的功能，用它可以求解非线性代数方程. 为了实现纠错码译码，首先需要确定错位的个数，然后构造译码方程并通过消元得到错位和错值的单变元方程. [11] 将这两步合二为一，采用变元提升的方法进行计算. 首先假定没有错误，检查所有伴随多项式是否为 0. 若为 0, 则说明接收数据正确，不用纠错；否则，假设有 i(i 从 1 开始) 位错误，构造译码方程组并计算对应多项式理想的 Gröbner 基，如果 Gröbner 基等于 1, 说明与实际错位数不符；然后，在此基础上假设有 $i+1$ 位错误，重新构造方程计算 Gröbner 基. 这样下去，直至 Gröbner 基不为 1 为止，说明译码方程有解即错位存在. 此时，Gröbner 基中关于错位变元的单变元多项式就是错位多项式.

在变元提升法中，需要不断地计算字典序的 Gröbner 基并进行判断. 因此，该方法虽然可以译码，但是效率较低. 特别是，对于纠错能力较强的纠错码，因为随着变元的增多该方法的译码效率呈指数性降低. 考虑到结式方法是非线性代数方程组的高效求解工具，该方法已用于代数攻击问题等的研究 [8]. 本文提出了基于 Dixon 结式理论的纠错码译码方法. 接收到数据后，根据码的生成多项式和它的完全定义集 $J(C)$ 得到最大纠错能力时的伴随式矩阵并计算该矩阵的秩得到错位个数. 然后，根据确定的错位个数并利用伴随多项式和错位及错值的关系构造译码方程. 最后，利用 Dixon 结式同时消去译码方程中的多个变元得到关于错位的单变元多项式. 本方法直接根据错位数构造译码方程，不需要靠猜的方法逐个提升变元. 另外，计算结式不需要确定变元的序，只需隐藏一个变元即作为参数处理即可. 和 Gröbner 基译码方法相比，该方法在计算过程简单且更实用. 同时，在计算过程中，不需要将伴随式等参数作为变元处理. 同时，译码的计算复杂度也比 Gröbner 基方法低.

2　Dixon 结式

早在 1779 年，Bezout 就创立了一个计算两个单变元多项式结式的方法. Dixon 在 1908 年将其推广到三个二变的多项式系统 [9]. 该方法通过逐步引入 k 个异于多项式中变元的新变元，将 Dixon 结式的计算方法推广到 $k+1$ 个 k 变元多项式系统. 一般多项式系统的 Dixon 结式的整个计算过程 [9] 如下.

给定 $k+1$ 个 k 变元多项式组 PS:

$$PS:\begin{cases} p_1(x_1,x_2,\cdots,x_k),\\ p_2(x_1,x_2,\cdots,x_k),\\ \vdots\\ p_{k+1}(x_1,x_2,\cdots,x_k). \end{cases} \tag{1}$$

设 $\alpha_1,\alpha_2,\cdots,\alpha_k$ 是 k 个新变元，则如下行列式：

$$\triangle(x_1,\cdots,x_k,\alpha_1,\cdots,\alpha_k)=\begin{vmatrix} p_1(x_1,x_2,\cdots,x_k) & \cdots & p_{k+1}(x_1,x_2,\cdots,x_k)\\ p_1(\alpha_1,x_2,\cdots,x_k) & \cdots & p_{k+1}(\alpha_1,x_2,\cdots,x_k)\\ \vdots & \vdots & \vdots\\ p_1(\alpha_1,\alpha_2,\cdots,\alpha_k) & \cdots & p_{k+1}(\alpha_1,\alpha_2,\cdots,\alpha_k) \end{vmatrix} \tag{2}$$

是关于 x,α 的多项式，并且当 $x_i=\alpha_i$ 时，

$$\triangle(x_1,\cdots,x_k,\alpha_1,\cdots,\alpha_k)=0.$$

这说明 $(x_1-\alpha_1)(x_2-\alpha_2)\cdots(x_k-\alpha_k)$ 是 $\triangle(x_1,\cdots,x_k,\alpha_1,\cdots,\alpha_k)$ 的因子．故

$$\delta(x_1,\cdots,x_k,\alpha_1,\cdots,\alpha_k)=\frac{\triangle(x_1,\cdots,x_k,\alpha_1,\cdots,\alpha_k)}{(x_1-\alpha_1)(x_2-\alpha_2)\cdots(x_k-\alpha_k)} \tag{3}$$

是一个多项式，并称之为 $p_1,p_2,\cdots,p_{k+1}$ 的 Dixon 多项式．

定义 1 行列式 $\triangle(x_1,\cdots,x_k,\alpha_1,\cdots,\alpha_k)$ 的元素形成的对应矩阵为 Cancellation 矩阵．

将 Dixon 多项式看做 $\alpha_1,\alpha_2,\cdots,\alpha_k$ 的多项式，设共有 m 项，并将 $\alpha_1,\alpha_2,\cdots,\alpha_k$ 的各不同幂积的系数记为

$$c_i(x_1,x_2,\cdots,x_k),\qquad i=1,2,\cdots,m.$$

它是关于 $x_1,x_2,\cdots,x_k$ 的多项式，称之为 $p_1=0,\ p_2=0,\cdots,p_{k+1}=0$ 的 Dixon 导出方程组．易证，原方程组的解必定是 Dixon 导出方程组的解．

在导出方程组中，把其中出现的关于变元 $(x_1,x_2,\cdots,x_k)$ 的所有幂积按字典序 (或全幂序) 由大到小写为：$e_n,\cdots,e_2,e_1$. 将 Dixon 导出方程组就可以写为标准形式：

$$D\cdot\begin{pmatrix} e_n\\ \vdots\\ e_2\\ e_1 \end{pmatrix}=\begin{pmatrix} 0\\ \vdots\\ 0\\ 0 \end{pmatrix}, \tag{4}$$

其中，D 是 Dixon 导出方程组关于幂积 $e_n,\cdots,e_2,e_1$ 的 $m\times n$ 阶系数矩阵，称为多项式组 $p_1,\ p_2,\ \cdots,\ p_{k+1}$ 的 Dixon 矩阵，记为 DM.

当 Dixon 矩阵是非奇异的方阵时，它的行列式称为 PS 的 Dixon 结式．如果 Dixon 矩阵奇异或非方时，无法得到关于变元的有用信息而使 Dixon 结式的使用受到限制．

杨路等人提出了计算广义 Dixon 结式的 KSY 方法，并证明只要 Dixon 矩阵中有一列不能表示成其它列的线性组合，就可以从中找出它的一个最大非奇异的子矩阵，它的行列式就可以作为 Dixon 结式 [9,10]. 这个条件在大多数情况下是可以满足的，它大大扩展了 Dixon 结式的使用范围. 利用 Dixon 结式理论可以同时消去非线性方程组的多个变元. 同时，结式具有重要的性质，即当多项式系统的最高次数项的系数不为 0 时，这 $n+1$ 个 n 变元多项式具有公共零点的必要条件是结式等于 0.

利用结式对多项式方程组进行处理时，要求方程个数比变元多一个. 而译码过程是由 n 个 n 变元多项式方程构成的方程组，可以将其中一个变元隐藏起来，即作为参数处理，从而构成符合要求的系统. 需要说明的是，因令多项式系统等于 0, 即可构成方程组. 多项式系统的零点就是方程组的解. 因此，后面叙述中不再区分多项式系统和方程组.

3　纠错码的译码过程分析

编码数据传送到接收端之后，接收端必须正确地恢复原来的数据，即译码过程. 译码的实质就是求解非线性代数方程组. 方法不同，得到不同的译码算法. 假设 C 是 $[n,k,d]$ 循环码，生成多项式为 $g(x)$, F_{q^e} 是 F_q 的扩域且 $n=q^e-1$, 其包含 $g(x)$ 的所有根且 $\alpha\in F_{q^e}$ 是 n 次单位本原根.

定义 2　一个整数的剩余系 $\mathbb{Z}_n$ 的子集 J 称为定义集 (defining set), 如果

$$C=\big\{c(X)\in\mathbb{F}/(X^n-1)|\ \text{对于所有的 } j\in J \text{ 有 } c(\alpha^j)=0\big\}.$$

同时，码 C 的完全定义集 (complete defining set) $J(C)$ 定义为：

$$J(C)=\big\{j\in Z_n\mid \text{对于所有的 } c\in C \text{ 使得 } c(\alpha^j)=0 \text{ 成立}\big\}.$$

若 C 的定义集为 $J=\{j_1,j_2,\cdots,j_r\}$, C 有校验矩阵

$$H=\begin{bmatrix}1 & \alpha^{j_1} & \alpha^{2j_1} & \cdots & \alpha^{(n-1)j_1}\\ 1 & \alpha^{j_2} & \alpha^{2j_2} & \cdots & \alpha^{(n-1)j_2}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \alpha^{j_r} & \alpha^{2j_r} & \cdots & \alpha^{(n-1)j_r}\end{bmatrix}.$$

若发送的码字 $C(x)=q(x)g(x)$, 接收的 n 重为 $R(x)=C(x)+\mathbf{e}(x)$. 假设错误图样 $\mathbf{e}(x)=e_{n-1}x^{n-1}+e_{n-2}x^{n-2}+\cdots e_1x+e_0$, 若信道产生 t 个错误，则 $\mathbf{e}(x)=Y_tx^{l_t}+Y_{t-1}x^{l_{t-1}}+\cdots+Y_1x^{l_1}$.

由伴随式的定义可知：

$$\mathbf{s}=\mathbf{He^T}=\mathbf{HR^T}=\begin{bmatrix}1 & \alpha^{j_1} & \alpha^{2j_1} & \cdots & \alpha^{(n-1)j_1}\\ 1 & \alpha^{j_2} & \alpha^{2j_2} & \cdots & \alpha^{(n-1)j_2}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \alpha^{j_r} & \alpha^{2j_r} & \cdots & \alpha^{(n-1)j_r}\end{bmatrix}\begin{bmatrix}0\\ \vdots\\ Y_1\\ \vdots\\ Y_t\\ \vdots\end{bmatrix}$$

$$=\begin{bmatrix}Y_1(\alpha^{j_1})^{l_{i_1}}+Y_2(\alpha^{j_1})^{l_{i_2}}+\cdots+Y_t(\alpha^{j_1})^{l_{i_t}}\\ Y_1(\alpha^{j_2})^{l_{i_1}}+Y_2(\alpha^{j_2})^{l_{i_2}}+\cdots+Y_t(\alpha^{j_2})^{l_{i_t}}\\ \vdots\\ Y_1(\alpha^{j_r})^{l_{i_1}}+Y_2(\alpha^{j_r})^{l_{i_2}}+\cdots+Y_t(\alpha^{j_r})^{l_{i_t}}\end{bmatrix}=\begin{bmatrix}s_1\\ s_2\\ \vdots\\ s_r\end{bmatrix}. \tag{5}$$

令 $x_i=\alpha^{j_i}$, 则 $\mathbf{s}=\sum\limits_{i=1}^{t}Y_ix_i^j,\ j=1,\cdots,r.$ 再根据有限域的特征，有 $Y_m^q=Y_m,\ m=1,\cdots,m=1,\cdots,t.$ 同时，由有限循环群理论可知： $x_i^{n-1}=1,\ i=1,\cdots,t.$

综合上述讨论得到多项式方程组如下 [3,12]:

$$\begin{cases}f_j=\sum\limits_{i=1}^{t}Y_ix_i^j-\mathbf{s_j}, & j=1,\cdots,r,\\ h_m=Y_m^q-Y_m, & m=1,\cdots,t,\\ g_m=x_i^{n-1}-1, & i=1,\cdots,t.\end{cases} \tag{6}$$

其中， x_i 表示错误位置， Y_i 表示错误位置的值. 在二进制码中，由于错误图样中错误位置的值为 1, Y_i 不再为变量. 这样，线性码的译码问题最终转化为多项式系统的求解问题. 利用 Dixon 结式计算错位多项式时，可以从式 (6) 选择部分方程构成变元和方程个数相等的方程组，其余方程作为约束条件以简化计算. 因错误位数是有限的，因此该方程组解的个数是有限的. 因此，方程组对应的多项式系统一定是零维多项式系统. 计算出 x_i 和 Y_i 的值之后，便可知道错误位置及错误位置的值. 据此，得到错误图样 $\mathbf{e}$. 然后，根据 $\mathbf{c}=\mathbf{R}-\mathbf{e}$ 得到正确的译码.

4 译码的 Dixon 结式方法

4.1 错位个数的计算

前面的译码分析是在假定接收数据中错误个数已知的情况下. 但是，实际译码过程中错位的个数是未知的. [11] 采用变元提升的办法. 计算时，假设错误位数为 i (可以从 1 开始), 据此构造译码方程并利用 Gröbner 基计算. 采用 Gröbner 进行循环码的具体译码算法可参阅 [11]. 如果假设的错误位数和实际位数不一致，译码方程无解. Gröbner 基有个绝佳的性质，代数方程无解的充分必要条件是它对应多项式系统的生成理想的 Gröbner 基为 $\langle 1\rangle$. 这样，逐一增加假设的错误位数直至 Gröbner 基不为 $\langle 1\rangle$ 就可得到错位多项式. 结式也有类似的性质，可以先猜错位个数并构造方程然后计算结式的方法进

行处理. 可是，反复计算结式是一个非常耗时的过程. 为解决该问题，在译码之前首先计算错位个数，确定错位之后构造译码方程. 这样，只需计算一次结式就可得到错位多项式.

因为每种纠错码的性能是已知的，所以该码的最大纠错数是确定的，不妨设纠错能力为 t. 然后，根据 t 和接受到的数据计算伴随式矩阵

$$\begin{bmatrix} s_{m_0+t-1} & s_{m_0+t-2} & \cdots & s_{m_0} \\ s_{m_0+t} & s_{m_0+t-1} & \cdots & s_{m_1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m_0+2t-2} & s_{m_0+2t-1} & \cdots & s_{m_0+t-1} \end{bmatrix}, \tag{7}$$

其中， m_0 为 $J(c)$ 中的最小值. 可以证明，当真正错位数小于 t 时该矩阵非满秩 [2]. 另一方面，传输过程中究竟有多少位错而无法确知. 此时，假设有 v 个错误。若 $v<t$, 式 (6) 中第一个方程左侧为 $\sum\limits_{m=1}^{v} Y_m X_m^j$. 根据伴随式的定义，显然等式不能成立，故方程无解.

定理 1[11]　假定接收数据 R 发生 t 个错误且 $t \le (d-1)/2$, 那么当 $v<t$ 时系统 $S(s,v)$ 在 F_{q^e} 上无解；当 $v=t$ 时，不考虑置换性，则有惟一解对应着最小重量的错误向量满足伴随式方程；如果 $v>t$, 那么对于每一个 j, 系统有解 $X_1=\alpha^j$.

根据该定理和 Dixon 结式的性质可以得到如下推论.

推论　设接收数据中有 t 位错误，但在求解过程中假定有 v 位错误. 若 $v<t$ 则式 (6) 的 Dixon 结式为一不等于 0 的常数. 只有 $v \ge t$ 时， Dixon 结式才是关于错误位置变元 X_i 的多项式.

该推论的成立是显然的. 因为当多项式系统无零点时，结式等于一个常数值. 但是，另一方面假设的错误个数 $v>t$ 时方程仍会有解，此时可能造成译码错误. 这仅仅是理论上的可能. 因为实际译码中，我们认为接收码组中的错误个数不会超过纠错码的纠错能力.

根据该推论只能确定接收码组中的错位数是否达到了纠错能力. 但是，实际接收数据中的错误数是随机的. 因此，最为关心的问题是接收数据中到底有多少位错误且如何高效地确定错位个数及其位置. 可以推导出错误位数和矩阵 (6) 有如下关系：

定理 2　当纠错码中的错误数小于设计的纠错能力时，伴随式矩阵 (7) 的秩就是接收数据中错位的个数.

当接收数据中有 t 位错误时，矩阵为满秩矩阵，秩 t 等于错误位数. 主要证明错位小于纠错能力时的情况.

证　假设接收数据中有 i $(i<t)$ 位错误，分别是 $x_1, x_2, \cdots, x_i$. 如果这 i 位没有在一起，计算时可以通过交换将其移动到前 i 位. 这样， i 位之后的 x_m $(m>i)$ 均为 0, 在此之前的项全不为 0.

因不知道接收数据的错误位数，假设其中的错误位数达到纠错能力 t. 由于矩阵任一行中非零元素的个数 $i<t$, 根据线性代数的相关知识使得矩阵

$$F=\begin{bmatrix}1 & 1 & \cdots & 1\\ x_1 & x_2 & \cdots & x_t\\ \cdots & \cdots & \ddots & \cdots\\ x_1^{t-1} & x_2^{t-1} & \cdots & x_t^{t-1}\end{bmatrix}\to\begin{bmatrix}1 & 1 & \cdots & 1 & \\ x_1 & x_2 & \cdots & x_i & \\ \cdots & \cdots & \cdots & \cdots & \mathbf{0}\\ x_1^{i-1} & x_2^{x-1} & \cdots & x_i^{i-1} & \\ & & \mathbf{0} & & \end{bmatrix}$$

等式右边表示一个左上角 $i\times i$ 元素不为 0, 其余元素为 0 的矩阵.

进一步,

$$\begin{bmatrix}1 & 1 & \cdots & 1\\ x_1 & x_2 & \cdots & x_t\\ \vdots & \vdots & \ddots & \vdots\\ x_1^{t-1} & x_2^{t-1} & \cdots & x_t^{t-1}\end{bmatrix}\begin{bmatrix}Y_1x_1^{m_0} & & & \\ & Y_2x_2^{m_0} & & \\ & & \ddots & \\ & & & Y_tx_t^{m_0}\end{bmatrix}$$

$$\cdot\begin{bmatrix}1 & x_1 & \cdots & x_1^{t-1}\\ 1 & x_2 & \cdots & x_2^{t-1}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & x_t & \cdots & x_t^{t-1}\end{bmatrix}$$

$$\longrightarrow\begin{bmatrix}1 & 1 & \cdots & 1 & \\ x_1 & x_2 & \cdots & x_i & \\ \vdots & \vdots & \vdots & \vdots & \mathbf{0}\\ x_1^{i-1} & x_2^{x-1} & \cdots & x_i^{i-1} & \\ & & \mathbf{0} & & \end{bmatrix}\begin{bmatrix}Y_1x_1^{m_0} & & & & \\ & Y_2x_2^{m_0} & & & \\ & & \ddots & & \\ & & & Y_ix_i^{m_0} & \\ & & & & \mathbf{0}\end{bmatrix}$$

$$\cdot\begin{bmatrix}1 & x_1 & \cdots & x_1^{i-1} & \\ 1 & x_2 & \cdots & x_2^{i-1} & \\ \vdots & \vdots & \vdots & \vdots & \mathbf{0}\\ 1 & x_i & \cdots & x_i^{i-1} & \\ & & \mathbf{0} & & \end{bmatrix}$$

$$\longrightarrow\begin{bmatrix}\sum\limits_{m=1}^{i}Y_mx_m^{m_0} & \sum\limits_{m=1}^{i}Y_mx_m^{m_0+1} & \cdots & \sum\limits_{m=1}^{i}Y_mx_m^{m_0+m-1} & \\ \sum\limits_{m=1}^{i}Y_mx_m^{m_0+1} & \sum\limits_{m=1}^{i}Y_mx_m^{m_0+2} & \cdots & \sum\limits_{m=1}^{i}Y_mx_m^{m_0+i} & \mathbf{0}\\ \vdots & \vdots & \vdots & \vdots & \\ \sum\limits_{m=1}^{i}Y_mx_m^{m_0+m-1} & \sum\limits_{m=1}^{i}Y_mx_m^{m_0+m} & \cdots & \sum\limits_{m=1}^{i}Y_mx_m^{m_0+2m-1} & \\ & & \mathbf{0} & & \end{bmatrix}.$$

令最后所得矩阵为 M_i, 根据线性代数的知识可得 M_i 的秩不会大于 i. 另一方面， M_i 左上角的 $i\times i$ 子矩阵恰为有 i 位错误时计算错误位置时的矩阵. 当有 i 位错误且不考虑置换性时方程组有惟一解. 所以，上述子矩阵的秩只能是 i.

另外，由式 (5) 可知

$$s_{m_0+l}=\sum_{m=1}^{i}Y_m x_m^{m_0+l-1}.$$

因此，矩阵 M_i 就是式 (7) 进行初等变换得到的. 故定理得证.

利用该定理，我们不必再采用变元提升的方法而是通过计算矩阵的秩就可以直接确定接收码组中错位的个数.

4.2 错位值的计算

确定错位个数之后，根据式 (6) 可以得到关于错误位置和伴随式关系的代数方程. 然后，利用 Dixon 结式在有限域上进行消元得到任一错误位置变元的多项式即错位多项式. 下面论述 Dixon 结式计算错位多项式的详细过程.

设 q 元循环码，由生成多项式得到码元的定义集为 $\{j_1,j_2,\cdots,j_r\}$, 接收码组中有 i 位错误, 分别是 $x_1,\cdots,x_i$, 相应的错误值分别是 $y_1,\cdots,y_i$, 伴随多项式为 $s_1,\cdots,s_i$. 根据式 (6) 得到多项式系统，其中的第三部分 g_m 作为约束条件使用. 接收码组确定之后，根据式 (5) 计算出 s_k $(1\le k\le i)$, 故 s_k 在多项式系统中只是参数，而变元是 x_k,y_k. 显然，整个系统由 $2i$ 个含有 $2i$ 变元的非线性方程构成方程组. 一方面， Dixon 结式适用于研究 $n+1$ 个 n 变元构成的多项式系统的根与系数之间的关系. 另一方面，译码方程需要进行消元并首先得到一个含有单变元的方程. 在译码过程中，需要将一个关于错误位置的变元视为参数，式 (6) 转换为 $2i$ 个含有 $2i-1$ 变元的方程组. 因为方程组 (6) 关于变元是对称的，隐藏任一变元不会改变计算错误位置的结果. 所以，可以将任何一个变元作为参数处理，不妨将 x_i 看做参数. 为了和真正的参数 $s_{1\cdots i}$ 相区别，称 x_i 为隐藏参数. 根据 Dixon 结式的构造过程开始构造结式. 首先计算 Δ 和 δ.

$$\begin{aligned}&\triangle(x_{1\cdots i-1},y_{1\cdots i},\alpha_{1\cdots i-1},\beta_{1\cdots i})\\&=\begin{vmatrix}f_1(x_{1\cdots i-1},y_{1\cdots i}) & \cdots & f_i(x_{1\cdots i-1},y_{1\cdots i}) & h_1(y_1) & \cdots & h_i(y_i)\\ f_1(\alpha_1,x_{2\cdots i-1},y_{1\cdots i}) & \cdots & f_i(\alpha_1,x_{2\cdots i-1},y_{1\cdots i}) & h_1(y_1) & \cdots & h_i(y_i)\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ f_1(\alpha_{1\cdots i-1},y_{1\cdots i}) & \cdots & f_i(\alpha_{1\cdots i-1},y_{1\cdots i}) & h_1(y_1) & \cdots & h_i(y_i)\\ f_1(\alpha_{1\cdots i-1},\beta_1,y_{2\cdots i}) & \cdots & f_i(\alpha_{1\cdots i-1},\beta_1,y_{2\cdots i}) & h_1(\beta_1) & \cdots & h_i(y_i)\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ f_1(\alpha_{1\cdots i-1},\beta_{1\cdots i}) & \cdots & f_i(\alpha_{1\cdots i-1},\beta_{1\cdots i}) & h_1(\beta_1) & \cdots & h_i(\beta_i)\end{vmatrix}.\end{aligned}\tag{8}$$

由于 $x_i,s_{1\cdots i}$ 是参数，故未在 $f_{1\cdots i}$ 表示出来. 上式除以 $\prod\limits_{k=1}^{i-1}(x_k-\alpha_k)\prod\limits_{k=1}^{i}(y_i-\beta_i)$ 得到 δ. 将 δ 看做关于 $\alpha_{1\cdots i-1}$, $\beta_{1\cdots i}$ 的多项式，则具有下述形式：

$$\delta=\sum_{k=0}^{m}c_k(x_{1\cdots i-1},y_{1\cdots i})\alpha_1^{d_{k_1}}\cdots\alpha_{i-1}^{d_{k_{i-1}}}\beta_1^{e_{k_1}}\cdots\beta_i^{e_{k_i}},$$

其中， $m=(2i)!\prod_{j=1\cdots i-1}\deg(x_j)\prod_{l=1\cdots i}\deg(y_l)$， $c_k(x_{1\cdots i-1},y_{1\cdots i})$ 表示多项式的系数．因为 f,g 含有参数 $s_{1\cdots i}$ 和 x_i, 所以 c_k 中也有这些参数．

考虑系数项 $c_k(x_{1\cdots i-1},y_{1\cdots i})$, 将其重新看做错位和错值变元 $x_{1\cdots i-1},y_{1\cdots i}$ 的多项式, c_k 又可以表示为：

$$c_k(x_{1\cdots i-1},y_{1\cdots i})=\sum_k c'_k(x_i,s_{1\cdots i})x_1^{d'_{k_1}}\cdots x_{i-1}^{d'_{k_{i-1}}}y_1^{e'_{k_1}}\cdots y^{e'_{k_i}}.$$

$\sum$ 下标 k 表示 $\sum_{j=1}^{i-1}d'_{k_j}+\sum_{j=1}^{i}e'_{k_j}=k$. 这样， $c_k(x_{1\cdots i-1},y_{1\cdots i})$ 是关于 $x_{1\cdots i-1},y_{1\cdots i}$ 的多项式， c_k 中的各个系数项 c'_k 也是多项式．但是， c'_k 是关于 $x_i,s_{1\cdots i}$ 的多项式．所有 c_k 中的 c'_k 按照式 (4) 构成矩阵，计算该矩阵行列式的值得到 Dixon 结式 $e_{\rm loc}$. 显然， $e_{\rm loc}$ 是关于 $x_i,s_{1\cdots i}$ 的多项式．因为 x_i 是隐藏参数， $s_{1\cdots i}$ 为真正参数．由此，我们得到关于错误位置的单变元多项式．根据结式理论，令 $e_{\rm loc}$ 等于 0 并求解可得到错误位置．顺便指出，因编译码是在有限域上进行，使用 Chien 搜索可以快速找到方程的解得到错误位置.

注 1 从形式上看，要得到所有 i 个错误位置需要计算 i 次 Dixon 结式或得到关于错位的三角型方程组．实际上，计算一次 Dixon 结式即可得到所有错位．如果接收码组有 i 个错误，得到的 Dixon 结式是一个关于任一变元 x_k 的 i 次多项式．求解该多项式可得到 i 个零点，即错误位置．不考虑置换性，方程组 (6) 的解是唯一的.

注 2 在构造错误位置的 Dixon 结式时，有时 Dixon 矩阵奇异或非方．这种情况确实存在，此时可用该矩阵的最大非奇异子式做为 Dixon 结式．相关理论可参阅 [11].

综合上述分析，得到循环纠错码的译码算法为：

算法 1

输入：接收到的数据 $\mathbf{y}$.

输出：正确的码字 C.

步骤：

1. 计算 $\mathbf{s}=\mathbf{y}H^T$.
2. 若对于所有 $j\in J$ 有 $\mathbf{s}^j=0$ 则 $\mathbf{y}$ 就是正确的码字，输出并结束程序;
   否则，利用伴随多项式构造矩阵 (7) 并计算矩阵的秩 (有限域) 确定错位个数 v.
3. 令 $PS=f_j\cup h_m,\ k=1,\cdots,v$..
4. 在 PS 中隐藏变元 x_v, 构造 Dixon 结式 $e_{\rm loc}$.
5. 利用 Chien 搜索，得到 $e_{\rm loc}$ 中 x_v 的解求出错误位置 $\mathbf{e}_1,\mathbf{e}_2,\cdots,\mathbf{e}_v$;
6. 代入方程 f_j, 求出错位处的错误值.
7. 根据 $\mathbf{c}=\mathbf{y}-\mathbf{e}$ 得到正确的码字并结束.

在译码算法中，最重要的就是 Dixon 结式的第 4 步．因为其它步骤或是线性计算或是代入验证，因此该步直接决定了译码的复杂度． Grönber 基译码的计算复杂度为 $(O(n+1)^{3t+1})$, 其中 n 是码长， t 是纠错能力 [5]; 而本算法的复杂度为 $(O(t^{3\sum_{i=1}^{t} i}))$. 在编码中，由于纠错能力 t 往往比 n 小很多，所以本算法的计算效率比 Gröbner 基方法的译码效率要低．后面的例子中给出了不同纠错能力的计算时间．

4.3 二元循环码的结式译码方法

在实际应用中，最常用的是二元编码．对于二元循环码， $q=2$, 错值总为 1. 这样，方程组 (6) 中的第二部分 h_m 不用考虑方程组可以进一步简化为 i 个 i 变元多项式的情形．同时，算法中也不用考虑第 7 步的错误值计算．

当编码中最多有两个错误时且假设 $\{1,3\}\subset J(C)$. 根据式 (6) 可以得到如下代数方程：

$$\begin{cases} x_1+x_2-s_1=0, \\ x_1^3+x_2^3-s_3=0. \end{cases} \tag{9}$$

方程中的其它等式作为约束条件在中间处理中使用．不考虑置换性，方程的解是唯一确定的．也就是说，当编码有两位错误且假定为 1 和 3 时，若 $x_1=1$ 则 $x_2=3$; 反之，若 $x_1=3$ 则 $x_2=1$. 因此，消去任意一个变元而得到关于另一个变元的方程不会影响编码的错误位置求解．计算错误位置时仅考虑最后得到的 Dixon 结式即可，不必将 Dixon 结式的零点代入到方程 (9) 计算另一个变元的值．计算时，不妨将变元 x_2 隐藏即将变元 x_2 做为参数处理．通过构造 Cancellation 矩阵得到 Dixon 多项式为：

$$\delta(x_1,\alpha_1)=-\alpha_1x_1^2-x_1\alpha_1^2-x_2x_1^2-x_2\alpha_1x_1-x_2\alpha_1^2+s_1x_1^2+s_1\alpha_1x_1+s_1\alpha_1^2+x_2^3-s_3.$$

式中 α_1 是为构造 Dixon 多项式，增加的与 x_1 相对应的新增变元．将 $\delta(x_1,\alpha_1)$ 看做关于变元 α_1 的多项式，整理得：

$$\delta(x_1,\alpha_1)=(s_1-x_2-x_1)\alpha_1^2+(s_1x_1-x_2x_1-x_1^2)\alpha_1+x_2^3+s_1x_1^2-x_2x_1^2-s_3.$$

可以看出， α_1^2,α_1^1,α_1^0 的系数分别为 $s_1-x_2-x_1, s_1x_1-x_2x_1-x_1^2, s_1x_1^2-x_2x_1^2-s_3$. 进一步，将上述系数重新看回为 x_2 的多项式并得到：

$$DM=\begin{bmatrix} {x_2}^3-s_3 & 0 & s_1-x_2 \\ s_1-x_2 & -1 & 0 \\ 0 & s_1-x_2 & -1 \end{bmatrix}.$$

矩阵 DM 中的元素是关于变元 x_2 的多项式．计算其行列式得到 Dixon 结式为 $e_{\mathrm{loc}}=s_3+s_1^3+s_1^2x_2+s_1x_2^2$. 根据结式理论可知，该单变元多项式在原多项式的生成理想中．因此，结式构成的代数方程的解就是原方程组中该变元的解．这样，最终得到错误位置多项式．因为式 (9) 是对称方程，消去 x_2 得到 x_1 的方程也具有同样的形式．所以，最后

得到任一变元的单变元多项式包含了错误位置的全部信息. 该方程的最高次数为 2, 说明方程有两个解, 即接收数据中有两位错误. 然后, 代入伴随式 s_1, s_3 的值并利用 Chien 搜索可得到错误位置. 因为是二元编码, 不需要求解错误值, 直接将错误位置的值置为 1. 最终利用 $\mathbf{c} = \mathbf{y} - \mathbf{e}$ 得到正确编码.

当编码中有 3 个错误且纠错能力为 3 时, $\{1, 3, 5\} \subset J(C)$. 由式 (6) 可以得到如下代数方程:

$$\begin{cases} x_1 + x_2 + x_3 - s_1 = 0, \\ x_1^3 + x_2^3 + x_3^3 - s_3 = 0, \\ x_1^5 + x_2^5 + x_3^5 - s_5 = 0. \end{cases} \tag{10}$$

将变元 x_1 隐藏, 利用 Dixon 结式的计算过程最后得到的就是的结式就是关于变元 x_1 的多项式, 即

$$e_{\mathrm{loc}} = (s_1 + x_1)^4((s_3 + s_1^3)x_1^3 + (s_1^4 + s_1 s_3)x_1^2 + (s_5 + s_3 s_1^2)x_1 + s_5 s_1 + s_3^2 + s_3 s_1^3 + s_1^6)^2.$$

Dixon 结式包括两部分, 第一部分为 $(s_1 + x_1)^4$. 如果有三位错误, $s_1 + x_1$ 的零点满足译码方程是不可能的. 因此, 该部分是多余因子. 式中第二部分就是错误位置多项式的平方, 去掉重根得到错位多项式. 整个计算过程利用 maple 8.0 在 DuoCore 1.6G CPU 的笔记本上的计算时间仅为 0.016 秒.

当编码的纠错能力达到 4 时, 若 $\{1, 3, 5, 7\} \subset J(C)$, 则译码方程为:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 - s_1 = 0, \\ x_1^3 + x_2^3 + x_3^3 + x_4^3 - s_3 = 0, \\ x_1^5 + x_2^5 + x_3^5 + x_4^5 - s_5 = 0, \\ x_1^7 + x_2^7 + x_3^7 + x_4^7 - s_7 = 0. \end{cases} \tag{11}$$

该方程组由 4 个四变元方程构成, 可以隐藏一个变元计算 Dixon 结式. 但是, 如果这样直接计算, 效率较低. 当变元较多时, 可以采用逐次计算结式的方法, 以在中间计算过程中除去多余因子和重因子. 计算时, 将方程组 (11) 中的第 1,2,3 个方程构成一个方程组, 第 1,2,4 个方程构成另一个方程组. 因每个方程组只有三个方程, 故只能消去两个变元, 不妨设消去变元 x_2, x_3. 最后, 得到两个含变元 x_1, x_4 的方程组. 第一个方程组得到的 Dixon 结式为:

$$\begin{aligned} e_{\mathrm{loc}_1} =& (s_1 + x_4 + x_1)^4(s_3 x_4^3 + s_3 s_1 x1^2 + s_3 s_1^2 x_1 + s_5 x_4 + s_5 x_1 + x_4^3 s_1^2 x_1 + x_4^3 s_1 x_1^2 + x_1^3 s_1^2 x_4 \\ &+ s_1 s_3 x_4^2 + s_1^2 s_3 x_4 + s_5 s_1 + x_1^3 s_3 + s_3^2 + s_1^3 s_3 + s_1^4 x_1^2 + x_1^3 s_1^3 + x_4^3 s_1^3 + s_1^4 x_4^2 + s_1^6 \\ &+ x_1^3 s_1 x_4^2 + s_3 x_4^2 x_1 + s_3 x_4 x_1^2). \end{aligned}$$

第 2 个方程组得到的 Dixon 结式为:

$$\begin{aligned} e_{\mathrm{loc}_2} =& (s_1 + x_1 + x_4)^2(s_3^2 x_4^2 x_1 + s_3 x_4^6 + s_3^2 x_4 x_1^2 + x_1^6 s_3 + s_7 x_4^2 + s_7 x_1^2 + s_3^3 + x_1^6 x_4^2 s_1 \\ &+ x_1^6 x_4 s_1^2 + x_4^6 x_1 s_1^2 + x_4^6 x_1^2 s_1 + x_1^6 s_1^3 + s_1^3 s_3^2 + s_1^8 x_1 + s_1^8 x_4 + x_4^6 s_1^3 + s_3^2 s_1 x_4^2 + s_3^2 s_1^2 x_1 \end{aligned}$$

$$+ s_3^2 s_1^2 x_4 + s_3^2 s_1 x_1^2 + s_1^9 + s_1^2 s_7)^2 (x_1^3 + s_3 + x_4^3).$$

在译码方程中 x_i $(1 \le i \le 4)$ 不可能等于 0, $(s_1 + x_4 + x_1)^4$ 的零点不是原多项式系统的零点，因此该项是 DM_1 中的多余因子；在 DM_2 中，$(s_1 + x_1 + x_4)^2$ 和 $(x_1^3 + s_3 + x_4^3)$ 是多余因子. 进一步，利用这两个方程消去变元消去 x_4 即得到含错误位置多项式的 Dixon 结式，结果为：

$$\begin{aligned} e'_{\mathrm{loc}} =& (s_3 + s_1^2 x_1 + s_1 x_1^2 + s_1^3)^5 (s_1^{10} + x_1^2 s_1^8 + s_1^7 s_3 + s_1^7 x_1^3 + s_1^6 s_3 x_1 + s_1^6 x_1^4 + s_1^5 s_3 x_1^2 + s_1^5 s_5 \\ &+ s_1^4 x_1^3 s_3 + s_1^4 s_5 x_1 + s_1^3 x_1^4 s_3 + s_7 s_1^3 + s_1^2 x_1^3 s_5 + s_1^2 s_5 s_3 + s_7 s_1^2 x_1 + s_1 x_1^3 s_3^2 \\ &+ s_1 s_5 x_1^4 + s_1 s_3^3 + s_7 s_1 x_1^2 + s_5^2 + s_3^3 x_1 + s_3 x_1^2 s_5 + x_1^4 s_3^2 + s_7 s_3)^3. \end{aligned}$$

去除多余因子和重因子，合并同类项得到错误位置多项式：

$$\begin{aligned} e_{\mathrm{loc}} =& (s_1^6 + s_5 s_1 + s_3^2 + s_1^3 s_3) x_1^4 + (s_1^7 + s_5 s_1^2 + s_1 s_3^2 + s_1^4 s_3) x_1^3 \\ &+ (s_3 s_1^5 + s_5 s_3 + s_1^8 + s_7 s_1) x_1^2 + (s_1^2 s_7 + s_3^3 + s_5 s_1^4 + s_1^6 s_3) x_1 \\ &+ s_1^{10} + s_1^2 s_5 s_3 + s_1^7 s_3 + s_1^5 s_5 + s_1 s_3^3 + s_5^2 + s_7 s_1^3 + s_7 s_3. \end{aligned}$$

整个计算过程仅需 0.184 秒完成.

在代数几何中，利用 Dixon 结式对多项式系统进行消元会产生多余因子，即其零点不是原多项式系统零点的因式项. 在实际计算中，需要计算机筛除多余因子和重因子. 分步计算的目的正在于此. 对于重因子，可以采用结式中的判别式方法去除. 多余因子问题是结式理论中的一个困难问题. 但是，在纠错码译码中可以根据实际问题能够判断出来并去除. 如果多余因子只含一个变元，则它的零点肯定不是译码方程的解. 在此情况下，可以用类似于 Chien 搜索的方法将其筛出. [13] 提出了一种去除多余因子的先验算法，据此方法可以去除含有多个变元的多余因子.

当错误位的个数确知的情况下，利用 Dixon 结式可以得到错误位置多项式. 但是，错位越多方程越复杂，求解过程也会越繁. 通过对二元域的错误位置多项式各项系数进行观察和分析，得到如下猜想：

猜想 所有符号如上所记，对于有 n 位错误的二元循环码，若 $\{1, 2, \cdots, 2n+1\} \subset J_c$，则错位多项式具有如下形式：

$$\mathrm{err\_locator} = \sum_{u=0}^{n} \prod (s_{i_m}^{j_m}) x^u i_1 j_1 + i_2 j_2 + \cdots + i_{2n+1} j_{2n+1} = \sum_{m=1}^{n} m - n$$

且 i_m, j_m 满足关系 $i_1 j_1 + i_2 j_2 + \cdots + i_{2n+1} j_{2n+1} = \sum\limits_{m=1}^{n} m - n.$

由此，可以很容易地得到 6 位错误时的错位多项式，而 [11] 利用计算机代数系统 Axiom 得到结果需要 4 个小时. x^i 各项的系数都是 s_{2m+1} 的多项式且项数分别是 $21, 21, 32, 46$. 各项的项数和 [11] 的计算结果是一致的. 利用该猜想推出 X^6 的系数项中每个单项式中伴随式上下脚标满足条件 $i_1 j_1 + i_2 j_2 + \cdots = \sum\limits_{m=1}^{6} m - n = 21 - 6 = 15$, 即

为：

$$s_1^{15}+s_1^{12}s_3+s_1^{10}s_5+s_1^8s_7+s_1^6s_9+s_1^4s_{11}+s_1^9s_3^2+s_1^5s_5^2+s_1s_7^2+s_1^6s_3^3+s_3^5+s_5^3$$
$$+s_1^2s_3s_5^2+s_1s_5s_9+s_3^2s_9+s_1^3s_5s_7+s_1^2s_3s_{11}+s_3s_5s_7+s_1s_3^3s_5+s_1^5s_3s_7.$$

因 [11] 没有给出 x^i 系数的具体值，所以还不能核实 5 位以上错误时该猜想的正确性. 如果该猜想正确，可以大大提高计算错误位置多项式的效率.

具体进行译码时，根据编码设计的代数结构和纠错能力采用结式方法计算出不同错位时的错误位置多项式. 这一步，可以用高性能计算机离线计算实现. 使用时，将不同错位时的错位多项式封装在芯片中. 这样，可以简化电路设计同时易于提高芯片的处理效率. 接收到数据后，生成伴随式矩阵 (7) 并计算它的秩. 然后，根据矩阵的秩调用相应的错误位置多项式. 求解时，使用 Chien 搜索计算以得到错误位置. 最后，根据错位计算出错位处的错误值.

5 实例分析

在本节用实例说明采用 Dixon 结式得到的错误位置多项式进行译码的具体过程.

例 1 设 α 是 $F_2[x]$ 中本原多项式 x^5+x^2+1 的一个根， C 是以 α 和 α^3 为根的二元 BCH 码 (码长 31), 如果收到 $\mathbf{y}=(1001011011110000110101010111111)$, 并且错误位置不超过 2 个，试得到正确译码 [1].

该问题可以采用其他方法得到正确译码. 在此，采用 Dixon 结式的方法进行译码. 因为 BCH 码是循环码，对多项式 $f=x^{31}-1$ 在 F_2 上进行因式分解，其有 7 个因式，即：

$$\begin{aligned}
f_1&=x^5+x^2+1;\\
f_2&=x^5+x^4+x^3+x+1;\\
f_3&=x^5+x^4+x^3+x^2+1;\\
f_4&=x^5+x^3+1;\\
f_5&=x^5+x^3+x^2+x+1;\\
f_6&=x^5+x^4+x^2+x+1;\\
f_7&=x+1.
\end{aligned}$$

因为 C 是以 α 和 α^3 为根的二元 BCH 码，所以它的码生成多项式 $g(x)=f_1f_3=x^{10}+x^9+x^8+x^6+x^5+x^3+1$. 另外， $\mathbf{y}$ 的多项式的对应形式为：

$$\begin{aligned}
r(x)=&1+x^3+x^5+x^6+x^8+x^9+x^{10}+x^{11}+x^{16}+x^{17}+x^{19}+x^{21}\\
&+x^{23}+x^{25}+x^{26}+x^{27}+x^{28}+x^{29}+x^{30}.
\end{aligned}$$

由定义 6 知该码的定义集为 $J=\{1,3\}$, 且知道错误位数不超过 2. 计算的伴随式：

$$s_1=r(\alpha)=\alpha^4+\alpha^2+\alpha;$$
$$s_3=r(\alpha^3)=1.$$

将 s_1,s_3 代入到错误位数为 2 的错误位置多项式中得：$\mathrm{err\_loc}=1+(a^4+a^2+a)^3+(a^4+a^2+a)^2x+(a^4+a^2+a)x^2$. 然后，在有限域 F_2^{31} 上进行 Chien 搜索 (Chien Search) 得到该多项式的解为 α^{14},α^{27}. 由此得出接收数据中第 17 位和第 4 位在传输过程中发生错误. 同样，通过消元得到关于 x_2 的单变元多项式方程并求解得到同样的结果. 该码是二元码，故错误图样的值为 1, 这样得到错误图样 $\mathbf{e}$. 然后，根据 $\mathbf{c}=\mathbf{y}-e$ 得到正确的码字. 因此，该接收数据中有两位错误. 当然，这仅仅是一个 Toy 模型，比较简单. 另外，该问题也可以用其他方法，如 Euclid 算法等，进行译码.

例 2　一个纠错能力为 3 的 [15,5,7]BCH 码. 它以 α,α^3,α^5 为根 (α 为本原多项式 x^4+x+1 的根) 发送了一个全 0 的码，接收数据多项式为 x^3+x^{10}. 试对其进行正确的译码.

因不知道错误个数，不能确定用那个错误位置多项式. 因此，首先需要确定错误个数，根据接收数据计算伴随多项式. 通过计算得到 $s_1=\alpha^{12},\ s_3=\alpha^7,\ s_5=\alpha^{10}$. 假定错误数达到了纠错能力 $t=3$, 构造矩阵 (7). 除了计算出 s_1,s_3,s_5 之外，还要计算 s_2,s_4, 分别为 α^9,α^3. 因此，矩阵 (7) 具有下述形式：

$$\begin{bmatrix} s_3 & s_2 & s_1 \\ s_4 & s_3 & s_2 \\ s_5 & s_4 & s_3 \end{bmatrix}=\begin{bmatrix} \alpha^7 & \alpha^9 & \alpha^{12} \\ \alpha^3 & \alpha^7 & \alpha^9 \\ \alpha^{10} & \alpha^3 & \alpha^7 \end{bmatrix}$$

对矩阵进行初等变换得到矩阵的秩为 2, 表明接收数据中有 2 位错误.

确定错误位数后，调用相应的错误位置多项式并代入相应的伴随式得到：$\mathrm{err\_loc}=\alpha^7+\alpha^6+\alpha^9x+\alpha^{12}x^2$. 在有限域上进行搜索得到 $x_1=\alpha^3,\ x_2=\alpha^{10}$. 根据错误位置和错误位置数的关系得到错误位置分别为 12 和 5. 然后，很容易的计算出错误图样并正确地译出发送数据为全 0 的码字.

6　结束语

错值多项式的计算是纠错码译码中的基本问题. 本文讨论了利用 Dixon 结式计算循环码的错值多项式. 在计算过程中，根据接收数据构造一个伴随式矩阵并利用伴随矩阵的秩判断错位个数. 然后，根据伴随式方程及有限域的特征等约束条件，将译码问题转换为多项式方程组的求解问题. 在译码过程中，利用 Dixon 结式对非线性方程组进行消元，得到含有错误位置变元 x_i 的单变元方程，利用 Chien 搜索求解该方程得到接收数据中的错误位置. 然后，将这些数据代入原方程组中求出错误值. 当错位较多时，方程组中的变元增多同时次数升高，从而使求解变得困难. 为此，根据已有的工作提出了关于错误位置的单变元方程的猜想. 该猜想在错误位数分别是 1,2,3,4 的情况下都是正

确的. 更多错误位数时的正确性需进一步证明. 利用该猜想可以大大提高译码效率. 另外, 针对编码计算都是在有限域上实现, 因此构造有限域上的 Dixon 结式构造方法是提高译码效率的另一条途径, 同时也是我们下一步研究的问题.

参 考 文 献

[1] 冯克勤. 纠错码的代数理论. 北京: 清华大学出版社, 2005
(Feng Keqin. The Algebraic Theory of Error-correcting Coding. Beijing: Tsinghua University Press, 2005)

[2] 王新梅, 肖国镇. 纠错码 - 原理与方法. 西安: 西安电子科技大学出版社, 2001
(Wang Xinmei, Xiao Guozhen. Error-correcting Coding-theory and Method. Xian: Xidian University Press, 2001)

[3] 陆佩忠. Gröbner 基与环上的线性递归阵列. 北京: 高等教育出版社, 2004
(Lu Peizhong. Gröbner Bases and Linear Recurring Arrays over Rings. Beijing: Higher Education Press, 2004)

[4] Fitzgerald J, Lax R F. Decoding Affine Variety Codes Using Groebner Bases. *Designs, Codes and Cryptography*, 1998, 13: 147–158

[5] Loustaunau P, York E V. On the Decoding of Cyclic Codes Using Gröbner Bases. *Applicable Algebra in Engineering, Communication and Computing*, 1997, 8: 469-483

[6] 李耀辉. 模的 Gröbner 基理论及在纠错码译码中的应用. 四川大学学报 (工程科学版), 2009, 41(1): 153–157
(Li Yaohui. Gröbner Bases Theory for Modules and Its Application in Decoding Error-correct Codes. *Journal of Sichuan University* (Engineering Science Edition), 2009, 41(1): 153–157)

[7] 李耀辉, 吴涛, 赵海豹等. 基于 Gröbner 基的纠错码译码方法. 武汉理工大学学报, 2010, 32(20): 51–55
(Li Yaohui, Wu Tao, Zhao Haibao, et al. Decoding Methods of Error-correct Codes by Using Gröbner Bases. *Journal of Wuhan University of Technology*, 2010, 32(20): 51–55)

[8] Cohen A M, Cuypers H, Hans S (Eds.) Some Tapas of Computer Algebra, Algorithms and Computation in Mathematics. Berlin: Springer-Verlag, 1999

[9] 唐懈瑾, 冯勇. Dixon 结式在密码学中的应用. 软件学报, 2007, 18(7): 1738–1745
(Tang Xijin, Feng Yong. Applying Dixon Resultant in Cryptography. *Journal of Software*, 2007, 18(7): 1738–1745)

[10] 杨路, 张景中, 侯晓荣. 非线性代数方程组及定理及机器证明. 上海: 上海科技出版社, 1996
(Yang L, Zhang J Z, Hou X R. Nonlinear Algebraic Equation System and Automated Theorem Proving. ShangHai: ShangHai Scientific and Technological Education Publishing House, 1996)

[11] Kapur D, Saxena T, Yang L. Algebraic and Geometric Reasoning Using Dixon Resultants. Proceedings of the International Symposium on Symbolic and Algebraic Computation, 1994

[12] Chen X, Reed I S, Helleseth T, Truong K. General Principles for the Algebraic Decoding of Cyclic Codes. *IEEE Transaction On Information Theory*, 1994, 40(5): 1661–1663

[13] 赵世忠，符红光. Dixon 结式的三类多余因子. 中国科学 (A 辑), 2008, 38(8): 949–960
(Zhao Shizhong, Fu Hongguang. Thress Kinds of Extraneous factors in Dixon Resultant. *Science in China* (Series A), 2008, 38(8): 949–960)

Decoding Cyclic Codes by Using Dixon Resultant Method

LI Yaohui ZHAO Haibao MA Chunya

(*Department of Computer Science, Tianjin University of Technology and Education, Tianjin* 300222)

(*E-mail: philliplee@163.com*)

Abstract In order to solve the nonlinear equations in the decoding of error-correcting codes, a new method, based on Dixon resultant, is proposed for computing the error locator polynomial. In the process of computation, a syndrome matrix should be generated according to error-correcting capability of the code and the received codeword firstly. Then, the number of errors in the received codeword is decided by the rank of a syndrome matrix. After that, the decoding equations can be constructed directly based on the number of errors and are eliminated by Dixon resultant, in which one of variables is regarded as a parameter. Finally, a univariate polynomial in the variable of error locator is computed out. When extraneous factors and repeated factors in the polynomial are deleted, the left is the error locator polynomial. Then error location can be known by using Chien search. If there are many errors in the received codeword, the system is divided into several subsystem for computing error locator polynomial by Dixon resultant step by step so as to improve the efficiency of algorithm. Thus, the extraneous factors and repeated factors in Dixon resultant can be sifted out and the computation is simplified. Besides this, from the error locator polynomial in different errors, a guess, which improves decoding efficiency greatly, of general error locator polynomial of a class of cyclic codes is proposed. However, its correctness should be proven theoretically. Contrast with other symbolic computation method, because the number of errors in codeword is decided directly and error locator polynomial can be computed out by using Dixon resultant only once. It can improve the efficiency of decoding and is benefit to simplify the design of decoding chip.

Key words Dixon resultant; error-correcting codes; decoding; syndrome
MR(2000) Subject Classification 04B35
Chinese Library Classification O236.2