Algebraic-Geometric Code and Modernised Algebraic Decoding

- Dr. Li Chen
  - Lecturer, School of Information Science and Technology, Sun Yat-sen University
  - BSc, MSc, PhD, MIEEE
  - Website: http://sist.sysu.edu.cn/~chenli
Personal Background

Education and employment
- 2003, BSc in Applied Physics, Jinan University, China
- 2004, MSc in Communications and Signal Processing, Newcastle University, UK
- 2008, PhD in Mobile Communications, Newcastle University, Supervisor: Prof. R. A. Carrasco (IET Fellow)
- 2007 – 2010, Research Associate, Newcastle University, engaged with an EPSRC project.
- 2010 -- .., Lecturer, Sun Yat-sen University

Research Interests
- Information theory and channel coding
- Cooperative system
Outline

- Part I - Algebraic-geometric codes
  - Construction of Hermitian Codes
  - Algebraic soft decoding of Hermitian codes
  - Performance evaluation (Hermitian vs. RS)
  - (Made in UK)

- Part II - Modernised algebraic decoding
  - Challenges → Inspiration
  - Modernisation: Progressive algebraic soft decoding (PASD)
  - Complexity reduction and performance evaluation
  - (Made in China)

- Conclusions and future work
1. Construction of Hermitian Codes

- Hermitian Curve: $H_w(x, y, z) = x^{w+1} + y^w z + yz^w$
  - Affine component: $H_w(x, y, 1) = x^{w+1} + y^w + y$ – used for code construction!

- Size of GF($q$) decides the degree of the curve: $w = \sqrt{q}$

- Genus of the curve: $g = w(w-1)/2$

- Designed distance of a $(n, k)$ Hermitian code: $d^* = n - k - g + 1$

- Size of the code: number of affine points $p_i = (x_i, y_i)$, $|p_i| = w^3 (> q)$

<table>
<thead>
<tr>
<th>Codes on fields</th>
<th>GF(4)</th>
<th>GF(16)</th>
<th>GF(64)</th>
<th>GF(256)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paras</td>
<td>deg</td>
<td>$g$</td>
<td>$n$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>28</td>
<td>512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>120</td>
<td>4096</td>
<td></td>
</tr>
</tbody>
</table>
I. Construction of Hermitian Codes

- Point of infinity $p_\infty$: for points that we can find in $H_w(1, y, z)$, $H_w(x, 1, z)$ and $H_w(x, y, 1)$, the one with the form of $(x_i, y_i, 0)$.
  - Variables $x, y, z$ have a pole order (or weights) at $p_\infty$, $x - w$, $y - w+1$, $z - ?$ (depends on $k$).

- Affine points $p_i$: points on an affine component. E.g. for $H_w(x, y, 1)$, $p_i$ satisfies $H_w(x_i, y_i, 1) = 0$.

- Pole basis $L_w$: a set of rational functions $\Phi_\alpha$ with increasing pole orders
  - Curve $H_2$ has $L_2 = \{1, x, y, x^2, xy, y^2, x^2y, xy^2, y^3, x^2y^2, xy^3, y^4, \ldots\}$
  - Curve $H_4$ has $L_4 = \{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, x^4, x^3y, x^2y^2, xy^3, y^4, x^4y, x^3y^2, x^2y^3, xy^4, y^5, \ldots\}$

- Zero basis $Z_{w,p_i}$: a set of rational functions $\psi_{w,p_i}$ with increasing zero orders at $p_i$. 
1. Construction of Hermitian Codes

- For a Hermitian code defined on the curve $H_w$:
  - Find out $n$ affine points on the curve – decide the length of the code
  - Select the first $k$ monomials in $L_w$ – decide the dimension of the code
  - With information symbols $(u_0, u_1, \ldots, u_{k-1}) \in \text{GF}(q)$, the message polynomial can be written as:
    $$u(x, y) = u_0 \Phi_0 + u_1 \Phi_1 + \ldots + u_{k-1} \Phi_{k-1}$$
  - And the codeword is generated by:
    $$(c_0, c_1, \ldots, c_{n-1}) = (u(p_0), u(p_1), \ldots, u(p_{n-1}))$$

- Example: Construct a (8, 4) Hermitian code defined over GF($2^2$)
  - Curve: $H_2 = x^3 + y^2 + y$
  - Affine points $p_0 = (0, 0)$, $p_1 = (0, 1)$, $p_2 = (1, \sigma)$, $p_3 = (1, \sigma^2)$, $p_4 = (\sigma, \sigma)$, $p_5 = (\sigma, \sigma^2)$, $p_6 = (\sigma^2, \sigma)$, $p_7 = (\sigma^2, \sigma^2)$.
  - Information symbols $1$, $\sigma$, $1$, $\sigma^2$, and message polynomial $u(x, y) = 1 + \sigma x + y + \sigma^2 x^2$.
  - Codeword $(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7) = (1, 0, \sigma, \sigma^2, \sigma, \sigma^2, \sigma^2, \sigma)$. 
## 1. A Comparison with RS Codes

<table>
<thead>
<tr>
<th>Properties</th>
<th>Codes</th>
<th>$(n, k)$ RS code</th>
<th>$(n, k)$ Hermitian code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebraic affine curves</td>
<td>$y = 0$</td>
<td></td>
<td>$x^{w+1} + y^w + y = 0$</td>
</tr>
<tr>
<td>Pole basis</td>
<td>$1, x, x^2, x^3, \ldots$</td>
<td>$1, x, y, x^2, xy, y^2, \ldots, x^w y, x^{w-1} y^2, \ldots, xy^w, y^{w+1}, \ldots$</td>
<td></td>
</tr>
<tr>
<td>Affine points ($p$)</td>
<td>$x_0, x_1, x_2, \ldots, x_{n-1}$</td>
<td>$(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_{n-1}, y_{n-1})$</td>
<td></td>
</tr>
<tr>
<td>Transmitted message polynomial ($u$)</td>
<td>$u(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_{k-1} x^{k-1}$</td>
<td>$u(x, y) = u_0 + u_1 \phi_1 + u_2 \phi_2 + \ldots + u_{k-1} \phi_{k-1}$</td>
<td></td>
</tr>
<tr>
<td>Codeword ($c$)</td>
<td>$(c_0, c_1, \ldots, c_{n-1}) = (u(x_0), u(x_1), \ldots, u(x_{n-1}))$</td>
<td>$(c_0, c_1, \ldots, c_{n-1}) = (u(p_0), u(p_1), \ldots, u(p_{n-1}))$</td>
<td></td>
</tr>
</tbody>
</table>
1. A Comparison with RS codes

- Advantage of AG codes: larger codes can be constructed from the same finite field as RS codes, resulting better error-correction capability;

- Example, over GF(64)

<table>
<thead>
<tr>
<th></th>
<th>Rate 0.3</th>
<th>Rate 0.56</th>
</tr>
</thead>
<tbody>
<tr>
<td>Herm (512, 153)</td>
<td>RS (63, 19)</td>
<td>Herm (512, 289)</td>
</tr>
<tr>
<td>d(^*) = 332</td>
<td>d = 45</td>
<td>d(^*) = 196</td>
</tr>
<tr>
<td>τ = 165</td>
<td>τ = 22</td>
<td>τ = 97</td>
</tr>
<tr>
<td>990 bits</td>
<td>132 bits</td>
<td>582 bits</td>
</tr>
</tbody>
</table>

- Disadvantage of AG codes: It is not a Maximum Distance Separable (MDS) code. Very high rate AG codes will be left with marginal error-correction capability.
1. A Comparison with RS codes

- AG vs. concatenated RS (512 ≈ 8 × 63)

8 RS
1 AG

- Complexity: $O(n^h)$

- Distribution of errors

- Diversity on codes
I. Overview of the algebraic decoding

- Decoding philosophy evolution

Unique decoding → List decoding

The Berlekamp-Massey algorithm
The Welch-Berlekamp algorithm
The Sakata algorithm with majority voting

The Guruswami-Sudan algorithm (Hard-decision)
The Koetter-Vardy algorithm (Soft-decision)

[Guruswami99], [Koetter03]
I. Overview of the algebraic decoding

- Key processes: Interpolation (construct $Q(x, y, z)$) + Factorisation (find out $u(x, y)$)

- From hard-decision decoding to soft-decision decoding ($GS \rightarrow KV$)
  
  **Hard-decision received word:** $\mathbf{R} = (r_0, r_1, \ldots, r_{n-1})$
  
  **Interpolated points:** $(p_0, r_0), (p_1, r_1), \ldots, (p_{n-1}, r_{n-1})$

  With certain multiplicity value $m$, perform:

  **Interpolation $Q(x, y, z)$** → **Factorisation $u(x, y)$**

  **Soft-decision reliability matrix $\Pi$ ($\rightarrow M$)**

<table>
<thead>
<tr>
<th>Encoding Channel</th>
<th>$p_0$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
<th>$p_5$</th>
<th>$p_6$</th>
<th>$p_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_0$</td>
<td>0.96</td>
<td>0.21</td>
<td>0.01</td>
<td>0.46</td>
<td>0.00</td>
<td>0.00</td>
<td>0.69</td>
<td>0.00</td>
</tr>
<tr>
<td>$r_1$</td>
<td>0.00</td>
<td>0.02</td>
<td>0.00</td>
<td>0.53</td>
<td>0.90</td>
<td>0.03</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$r_2$</td>
<td>0.03</td>
<td>0.74</td>
<td>0.03</td>
<td>0.01</td>
<td>0.10</td>
<td>0.02</td>
<td>0.28</td>
<td>0.99</td>
</tr>
<tr>
<td>$r_3$</td>
<td>0.01</td>
<td>0.03</td>
<td>0.94</td>
<td>0.00</td>
<td>0.00</td>
<td>0.95</td>
<td>0.03</td>
<td>0.00</td>
</tr>
</tbody>
</table>

  Where a multiplicity value $m_{ij}$ was assigned to the unit

  $(p_0, 0)$, $(p_1, 0)$, $(p_2, \sigma^2)$, $(p_3, 0)$, $(p_4, 1)$, $(p_5, \sigma^2)$, $(p_6, 0)$, $(p_7, \sigma)$, $(p_1, \sigma)$, $(p_3, 1)$, $(p_4, \sigma)$, $(p_6, \sigma)$
Algebraic soft decoding of Hermitian codes

- From RS to Hermitian: [Chen09], [Lee10]
  - Bivariate monomials (polynomials) $\rightarrow$ trivariate monomials (polynomials)
  - Define the interpolated zero conditions
    - Calculate the corresponding coefficients of a Hermitian curve
  - Validity of the algorithm
  - Optimal performance bound
  - Complexity reduction methods
1. Trivariate monomials (Polynomials)

- For a code defined on the curve \( H_w = x^{w+1} + y^w + y \),
  - monomial \( x^i y^j z^k \), \( 0 \leq i \leq w, j \geq 0 \) and \( k \geq 0 \)
  - Decoding a \((n, k)\) Hermitian codes, \( \deg_w(z) = \deg_w(\Phi_{k-1}) \)
  - \( \deg_w(x^i y^j z^k) = iw + j(w+1) + k\deg_w(z) \)
  - For to monomials \( x^i_1 y^j_1 z^k_1 \) and \( x^i_2 y^j_2 z^k_2 \)
    \[ x_1^i y_1^j z_1^k < x_2^i y_2^j z_2^k \]
    - if \( \deg_w(x_1^i y_1^j z_1^k) < \deg_w(x_2^i y_2^j z_2^k) \), or \( \deg_w(x_1^i y_1^j z_1^k) = \deg_w(x_2^i y_2^j z_2^k) \) and \( k_1 < k_2 \).
  - A lexicographic order can be assigned to monomials.

- Polynomials \( Q(x, y, z) = \sum_{a, b \in N} Q_{ab} \phi_a(x, y) z^b \), \( Q_{ab} \in \text{GF}(q) \)
  - Identify the maximal monomial in \( Q(x, y, z) \) as \( \phi_a z^b \), then \( \deg_w(Q) = \deg_w(\phi_a z^b) \)
  - Leading order, \( \text{lod}(Q) = \text{ord}(\phi_a z^b) \)

- \( N_w(\delta) = |\{ \phi_a z^b : \deg_w(\phi_a z^b) \leq \delta, (a, b, \delta) \in N \}| \) Define the number of monomials

- \( \Delta_w(v) = \min\{ \delta : N_w(\delta) > v, v \in N \} \) Define the weighted degree of monomials
1. Define the Interpolated Zero Conditions

- To interpolate unit \((p_i, r_i)\) (or \((x_i, y_i, r_i)\))
- Recall the zero basis \(Z_{w,pi}\) with rational functions \(\psi_{pi,\alpha}\) as:
  \[
  \psi_{p_i,\alpha} = \psi_{p_i,\lambda+(w+1)\delta} = (x - x_i)^\lambda [(y - y_i) - x_i^w (x - x_i)]^\delta, \quad (0 \leq \lambda \leq w, \delta \geq 0)
  \]
- Zero condition with multiplicity \(m\) for polynomial \(Q(x, y, z) = \sum_{a,b \in N} Q_{ab} \phi_a (x, y) z^b\)
  - It can be written as: \(Q(x, y, z) = \sum_{\alpha, \beta \in N} Q_{\alpha\beta}^{(p_i, r_i)} \psi_{p_i,\alpha} (z - r_i)^\beta\)
  - \(Q_{\alpha\beta}^{(p_i, r_i)} = 0\) for \(\alpha + \beta < m\).
- Since \(\phi_a = \sum_{\alpha \in N} \gamma_{a,p_i,\alpha} \psi_{p_i,\alpha}\) and \(z^b = \sum_{\beta \leq b} \binom{b}{\beta} r_i^{b-\beta} (z - r_i)^\beta\)

  \[
  Q_{\alpha\beta}^{(p_i, r_i)} = \sum_{a,b \geq \beta} Q_{ab} \binom{b}{\beta} \gamma_{a,p_i,\alpha} r_i^{b-\beta} \quad [Nielsen01]
  \]

A key parameter for determining the polynomial’s zero condition!
1. Calculate the Corresponding Coefficients

- Lemma: \( \phi_a = \sum_{\alpha \in N} \gamma_{a,p_t,\alpha} \psi_{p_t,\alpha} \leftrightarrow \psi_{p_t,\alpha} = \sum_{\alpha \in N} \zeta \phi_a \), \( \psi_{p_t,\alpha} = \sum_{\alpha \in N, a < L} \zeta \phi_a + \phi_L \).

- Recursive corresponding coefficient search algorithm [Chen08]

**Algorithm A:** Determining the corresponding coefficients \( \gamma_{a,p_t,\alpha} \) between a pole basis monomial \( \phi_a \) and zero basis functions \( \psi_{p_t,\alpha} \).

**Step 1:** Initialise all corresponding coefficients \( \gamma_{a,p_t,\alpha} = 0 \);

**Step 2:** Find the zero basis function \( \psi_{p_t,\alpha} \) with \( LM(\psi_{p_t,\alpha}) = \phi_a \), and let \( \gamma_{a,p_t,\alpha} = 1 \);

**Step 3:** Initialise function \( \hat{\psi} = \psi_{p_t,\alpha} \);

**Step 4:** While \( \hat{\psi} \neq \phi_a \) {

**Step 5:** Find the second largest pole basis monomial \( \psi_{L-1} \) with coefficient \( \zeta_{L-1} \) in \( \hat{\psi} \);

**Step 6:** In \( Z_{w,p_t} \), find a zero basis function \( \psi_{p_t,\alpha} \) whose leading monomial \( LM(\psi_{p_t,\alpha}) = \phi_{L-1} \), and let the corresponding coefficient \( \gamma_{a,p_t,\alpha} = \zeta_{L-1} \);

**Step 7:** Update \( \hat{\psi} = \hat{\psi} + \gamma_{a,p_t,\alpha} \psi_{p_t,\alpha} \);

}
1. Validity of the Algorithm

- **Condition 1:** From the perspective of solving a linear equation group
  \[ N_w(\delta) > C_M \]
  \[ \text{Freedom (Nr of coefficients)} \quad \text{Constraints} \]

- **Condition 2:** From the perspective of solving equation \( Q(x, y, u) = 0 \)
  \[ S_M(C) > \deg_w(Q(x, y, z)) \]
  \[ \text{Total zero order of } Q \quad \text{Pole order of } Q \]

- **Theorem 2:** Given the multiplicity matrix \( M \) and the resulting interpolated polynomial \( Q(x, y, z) \), if the codeword score \( S_M(C) \) is large enough such that:
  \[ S_M(C) > \deg_w(Q(x, y, z)) \]
  message polynomial \( u \) can be found out by factorising \( Q \) as:
  \[ z - u \mid Q(x, y, z) \]
  or \( Q(x, y, u) = 0. \) → This gives a tight condition of successful list decoding!!!

[Chen09]
1. Prove the Validity of the Algorithm

- A corollary that can embrace both of the successful decoding conditions.

**Corollary 3:** Message polynomial $f$ can be found out by $z - u \mid Q(x, y, z)$ if
$$S_M(C) > \Delta_w(C_M)$$

Since $\Delta_w(C_M)$ guarantees $N_w(\delta) > C_M$ (Condition 1 is met!)

Since $\deg_w(Q(x, y, z)) \leq \Delta_w(C_M)$, if $S_M(C) > \Delta_w(C_M)$, $S_M(C) > \deg_w(Q)$
(Condition 2 is met!)

**Remark:** Solving the linear polynomial group does not give a tight bound on successful list decoding, but solving the polynomial $Q(x, y, u) = 0$ does!

This can be seen later.
1. Optimal Performance Bound

- **Corollary 4**: Let \( w_z = \deg_w(\Phi_{k-1}) \), \( N_w(\delta) > \delta(\delta - g)/2w_z \) given \( \delta > 2g - 1 \). And \( N_w(\delta) = \delta^2/2w_z \) with \( \delta \to \infty \).

- With \( l \to \infty \), algebraic soft decoding algorithm’s asymptotic optimal performance can be achieved.

\[
l \to \infty, \ C_M \to \infty \text{ and } \Delta_w(C_M) \to \infty, \text{ it results } \Delta_w(C_M) \approx \sqrt{2w_zC_M}
\]

- Corollary 3 \( (S_M(C) > \Delta_w(CM)) \) can be interpreted as:

\[
\sum_{j=0}^{n-1} \bar{m}_{i,j} > \sqrt{w_z \sum_{i=0}^{q-1} \sum_{j=0}^{n-1} m_{i,j}(m_{i,j} + 1)}.
\]

[Chen09]
1. Optimal Performance Bound

- Asymptotic condition (when $C_M \to \infty$): \[
\frac{\pi_{i,j}}{n} = \frac{m_{i,j}}{s}
\]

- We could further have

- Since with $s \to \infty$, $n/s \to 0$ and

\[
\sum_{j=0}^{n-1} \frac{s}{n} \pi_{i,j} > \frac{s}{n} w_z \sum_{i=0}^{q-1} \sum_{j=0}^{n-1} \pi_{i,j} \left( \pi_{i,j} + \frac{n}{s} \right).
\]

- In KV decoding of RS codes, $w_z$ is replaced by $k - 1$.

- The performance of the KV algorithm is bounded by the quality of the received information $\Pi$.

- Had the quality of $\Pi$ been improved, optimal performance bound can be enhanced. [El-Khamy06]
Complexity Reduction Methods

- Modified reliability transform algorithm (introducing a stopping criterion) [Chen09]
  - In KV, reliability transform is stopped once a predefined $s = \sum_{i,j} m_{i,j}$ is met.
  - Reliability transform is stopped once a predefined output list size $l$ is met.

- Pre-calculation of the corresponding coefficients [Chen08]
  - Determine $\gamma_{a,p_i,\alpha}$

- Elimination of the unnecessary polynomials in the group [Chen07]
  - Eliminate polynomials with $\text{lod}(Q) > C_M$
1. Complexity reducing interpolation

- Pre-calculation of the corresponding coefficients and elimination of the unnecessary polynomials

\[ G = \{ Q \mid D(Q) = 0 \} \]

After \( C \) iterations, output \( Q^* \)

For \( Q \) with \( D(Q) \neq 0 \)

Eliminate polynomials with \( lod \) over \( C \)

Bilinear modifications

In the end, the minimal polynomial \( Q \) in group \( G \) is chosen!
1. Complexity reducing interpolation

The (64, 19) Hermitian code

The graph shows the computation complexity for both original interpolation and complexity reducing interpolation for different values of $E_b / N_0$ in dB. The percentage of complexity is compared for $l = 2$ and $l = 3$. The graph indicates a significant reduction in complexity for the complexity reducing interpolation method.
Arising Awareness

- Why Condition 1 \((N_w(\delta) > C_M)\) is NOT a tight bound?

- Since \(\text{lod}(Q^*) \leq C_M\), if \(\text{deg}_w(Q^*) = \delta^*\), then

\[
N_w(\delta^*) \leq C_M \quad \text{and} \quad N_w(\delta) > C_M
\]

- \(N_w(\delta) > C_M\) is the successful decoding criterion w.r.t. the polynomial group \(G\). However, the minimal polynomial in \(G\) does not meet this condition.

- To access the decoding performance, only Condition 2 gives a tight bound:

\[
S_M(\ M( \ ) \ > \ deg_w(Q(x, y, z))
\]

- Since \(\text{deg}_w(Q(x, y, z)) \leq \Delta_w(C_M)\), without performing the interpolation process, the theoretical assessment (e.g. \(S_M(\ M( \ ) \ > \Delta_w(C_M))\)) produces a relatively negative results.
Performance Evaluation

Hermitian code (512, 289) over AWGN channel

I is the output list size of the list decoder

.... approaching the optimal bound!

.... performance advantage of the list decoding algorithms!
1. Hermitian code ~ RS code

Both codes are defined in GF(64), over AWGN channel

- Hermitian (512, 289), soft-decision ($l = 1$)
- Hermitian (512, 289), soft-decision ($l = 2$)
- Hermitian (512, 289), soft-decision ($l = 5$)
- RS (63, 35), soft-decision ($l = 1$)
- RS (63, 35), soft-decision ($l = 2$)
- RS (63, 35), soft-decision ($l = 5$)

### Table: Performance Comparison

<table>
<thead>
<tr>
<th>Output size</th>
<th>Codes</th>
<th>Hermitian (512, 289)</th>
<th>RS (63, 35)</th>
<th>RS (255, 144)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l = 1$</td>
<td>$C = 892$</td>
<td>$C = 103$</td>
<td>$C = 430$</td>
<td></td>
</tr>
<tr>
<td>$l = 2$</td>
<td>$C = 1813$</td>
<td>$C = 204$</td>
<td>$C = 859$</td>
<td></td>
</tr>
<tr>
<td>$l = 5$</td>
<td>$C = 4602$</td>
<td>$C = 715$</td>
<td>$C = 3004$</td>
<td></td>
</tr>
</tbody>
</table>
1. Hermitian code ~ RS code

Hermitian code is defined in GF(64) and RS code is defined in GF(256)

<table>
<thead>
<tr>
<th>Output size</th>
<th>Codes</th>
<th>$C_{(512, 289)}$</th>
<th>$C_{(63, 35)}$</th>
<th>$C_{(255, 144)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l = 1$</td>
<td>$C = 892$</td>
<td>$C = 103$</td>
<td>$C = 430$</td>
<td></td>
</tr>
<tr>
<td>$l = 2$</td>
<td>$C = 1813$</td>
<td>$C = 204$</td>
<td>$C = 859$</td>
<td></td>
</tr>
<tr>
<td>$l = 5$</td>
<td>$C = 4602$</td>
<td>$C = 715$</td>
<td>$C = 3004$</td>
<td></td>
</tr>
</tbody>
</table>

$E_b / N_0$ [dB] vs. BER plot showing performance comparison for different codes and soft-decision levels.
II. Modernised algebraic decoding

- Challenges → Inspirations
- Modernisation: Progressive algebraic soft decoding (PASD)
- Complexity reduction and performance evaluation
II. Challenges → Inspirations

- The algebraic soft decoding is of high complexity, mainly due to the iterative interpolation process.
- A rebound thinking – a common phenomenon for most of the modern decodings.

**Inspiration:** Can we design an algebraic decoder which can also adjust its complexity according to the quality of the received word?

We can ‘borrow’ the idea from iterative decoding!
II. Challenges → Inspirations

- A review towards the modern codes (LDPC or Turbo codes)
  - The Belief Propagation (BP) algorithm with a parity check matrix $H$

\[ \text{Yes, } \hat{C} \]
\[ \hat{C}H^T = 0? \]
\[ \text{No} \]

- An iterative process
- Incremental computations between iterations
- A continue test of the decoding output
- Decoding capability and complexity can be adjusted according to the quality of $\mathcal{R}$
II. Modernised algebraic decoding

- The existing complexity reduction approaches
  - Facilitated reliability transform: $M = [\lambda \cdot \Pi]$ [Gross06]
  - Coordinate transform: \{$(\alpha_0, y_0), (\alpha_1, y_1), \ldots, (\alpha_{k-1}, y_{k-1}), (\alpha_k, y_k), \ldots, (\alpha_{n-1}, y_{n-1})$\}
  - Elimination of unnecessary polynomials: $G = \{Q \mid \text{lod}(Q) \leq C_M\}$ [Chen07]
  - Hybrid decoding:

\[
\text{BM} \quad \begin{array}{c}
\text{yes} \\
\text{no}
\end{array} \quad \text{ASD (KV)} \quad \hat{C} \quad \hat{C} \quad \text{BM} \quad \begin{array}{c}
\text{yes} \\
\text{no}
\end{array} \quad \text{ASD (KV)} \quad \hat{C}
\]

\text{BM} \quad \begin{array}{c}
\text{yes} \\
\text{no}
\end{array} \quad \text{ASD (KV)} \quad \hat{C} \quad \hat{C} \quad \text{BM} \quad \begin{array}{c}
\text{yes} \\
\text{no}
\end{array} \quad \text{ASD (KV)} \quad \hat{C}
\]
## Construction of a \((n, k)\) RS Code

### The message polynomial evaluation

- Let \(u = (u_0, u_1, \ldots, u_{k-1}) \in \text{GF}(q)\) be a message vector, forming a message polynomial:

\[
u(x) = u_0 + u_1 x + \cdots + u_{k-1} x^{k-1}\]

- Choosing \(n\ (n \leq q)\) distinct elements \(\alpha_0, \alpha_1, \ldots, \alpha_{n-1} \in \text{GF}(q)\\{0\},\) the output codeword \(c\) can be generated as

\[
c = (c_0, c_1, \ldots, c_{n-1}) = (u(\alpha_0), u(\alpha_1), \ldots, u(\alpha_{n-1}))\]
II. A graphical thinking

If $c_6$ is the transmitted codeword, PASD completes the decoding with $l = 1$ rather than $l = 5$ as the KV algorithm – optimizing the assignment of decoding parameters & complexity.
\[ \Pi \xrightarrow{\begin{array}{c} l_1 \\ l_v \\ \text{No, } v = v + 1 \end{array}} \text{M}_v \xrightarrow{\text{Inpolation } Q(x,y)} \text{Factorization } p(x), L \]

\[ u(x) \in L? \xrightarrow{\begin{array}{c} \text{Yes} \\ \text{No} \end{array}} \text{Output } u(x) \]

\[ l_v \text{- designed output list size at each iteration;} \]
\[ l_{\max} \text{- the designed maximal output list size;} \]
\[ l' \text{- step size for updating the output list size;} \]
\[ L \text{- the output list of all polynomials } p(x) \text{ such that } y-p(x)|Q(x, y). \]

Two key steps: Progressive Reliability Transform (PRT) \( \Rightarrow \) \( M_1, M_2, \ldots, M_v, \ldots \)
Progressive Interpolation (PIP) \( \Rightarrow \) \( Q_1(x, y), Q_2(x, y), \ldots, Q_v(x, y), \ldots \)

[Tang11]
II. Defining the zero condition constraints

- Multiplicity $m_{ij}$ ~ interpolated point $(x_j, \alpha_i)$
- Given a polynomial $Q(x, y)$, $m_{ij}$ implies $D_{r,s}(Q(x, y))|_{x=x_j, y=\alpha_i} = 0$ for $r + s < m_{ij}$

**Definition 1**: Let $\Lambda(m)$ denotes a set of zero condition constraints $(r, s)$ indicated by $m$, then $\Lambda(M)$ denotes a collection of all the sets $\Lambda(m_{ij})$ defined by the entry $m_{ij}$ of $M$

$$\Lambda(M) = \{\Lambda(m_{ij}), m_{ij} \in M\}$$

- **Example**: 

  $$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  $$\Lambda(M) = \{\{(0, 0), (1, 0), (0, 1)\}_{00}, \emptyset_{01}, \emptyset_{02}, \emptyset_{10}, \{(0, 0)\}_{11}, \{(0, 0)\}_{12}, \{(0, 0)\}_{20}, \{(0, 0), (1, 0), (0, 1)\}_{21}, \emptyset_{22}, \emptyset_{30}, \emptyset_{31}, \{(0, 0)\}_{32}\}$$
II. Defining the zero condition constraints

- **Definition 2:** Let \( m_{ij}^v \) and \( m_{ij}^{v+1} \) denote the entries of matrix \( M_v \) and \( M_{v+1} \), the incremental zero condition constraints introduced between the matrices are defined as a collection of all the residual sets between \( \Lambda(m_{ij}^{v+1}) \) and \( \Lambda(m_{ij}^v) \) as:

\[
\Lambda(\Delta M_{v+1}) = \Lambda(M_{v+1}) - \Lambda(M_v) = \{\Lambda(m_{ij}^{v+1}) - \Lambda(m_{ij}^v)\}
\]

- **Example:**

\[
M_2 = \begin{bmatrix}
2 & 0 & 0 \\
0 & 1 & 1 \\
1 & 2 & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad \quad
M_1 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
\Lambda(M_2) = \{(0, 0), (1, 0), (0, 1)\}_{00}, \emptyset_{01}, \emptyset_{02}, \emptyset_{10}, \{(0, 0)\}_{11}, \{(0, 0)\}_{12}, \{(0, 0)\}_{20}, \{(0, 0), (1, 0), (0, 1)\}_{21}, \emptyset_{22}, \emptyset_{30}, \emptyset_{31}, \{(0, 0)\}_{32}\}
\]

\[
\Lambda(M_1) = \{(0, 0)\}_{00}, \emptyset_{01}, \emptyset_{02}, \emptyset_{10}, \emptyset_{11}, \{(0, 0)\}_{12}, \{(0, 0)\}_{20}, \{(0, 0)\}_{21}, \emptyset_{22}, \emptyset_{30}, \emptyset_{31}, \{(0, 0)\}_{32}\}
\]

\[
\Lambda(\Delta M_2) = \{(1, 0), (0, 1)\}_{00}, \emptyset_{01}, \emptyset_{02}, \emptyset_{10}, \{(0, 0)\}_{11}, \emptyset_{12}, \emptyset_{20}, \{(1, 0), (0, 1)\}_{21}, \emptyset_{22}, \emptyset_{30}, \emptyset_{31}, \emptyset_{32}\}
\]

10 constraints

5 constraints

5 constraints
II. Progressive Interpolation

- PIP (Λ(M), G) – the Interpolation process that involves a group of polynomials G with respect to constraints of Λ(M).

- \( M_1, M_2, M_3, \ldots, M_{v-1}, M_v, \ldots, M_{max-1}, M_{max} \)

\[
\begin{align*}
\Lambda(M_1), & \quad \Lambda(M_2), & \quad \Lambda(M_3), & \quad \ldots, & \quad \Lambda(M_{v-1}), & \quad \Lambda(M_v), & \quad \ldots, & \quad \Lambda(M_{max-1}), & \quad \Lambda(M_{max}) \\
\Lambda(\Delta M_2), & \quad \Lambda(\Delta M_3), & \quad \ldots, & \quad \Lambda(\Delta M_v), & \quad \ldots, & \quad \Lambda(\Delta M_{max-1}), & \quad \Lambda(\Delta M_{max}) \\
G_1, & \quad G_2, & \quad G_3, & \quad \ldots, & \quad G_{v-1}, & \quad G_v, & \quad \ldots, & \quad G_{max-1}, & \quad G_{max} \\
\Delta G_1, & \quad \Delta G_2, & \quad \ldots, & \quad \Delta G_{v-1}, & \quad \ldots, & \quad \Delta G_{max-1}, & \quad \Lambda Q_M(x, y)
\end{align*}
\]
II. Progressive interpolation

- PIP ($\Lambda(M), G$) – the Interpolation process that involves a group of polynomials $G$ with respect to constraints of $\Lambda(M)$.

$$
\text{PIP}(\Lambda(M_1), G_1) + \\
\text{PIP}(\Lambda(M_1), \Delta G_1) + \text{PIP}(\Lambda(\Delta M_2), G_2) + \\
\text{IP}(\Lambda(M_{max}), G_{max}) \Rightarrow \\
\text{PIP}(\Lambda(M_2), \Delta G_2) + \text{PIP}(\Lambda(\Delta M_3), G_3) + \\
\vdots \\
\text{PIP}(\Lambda(M_{v-1}), \Delta G_{v-1}) + \text{PIP}(\Lambda(\Delta M_v), G_v) + \\
\vdots \\
\text{PIP}(\Lambda(M_{max-1}), \Delta G_{max-1}) + \text{PIP}(\Lambda(\Delta M_{max}), G_{max})
$$

- The number of ‘factorisations’ has been increased. However, its complexity is rather marginal compared to interpolation.
II. Implementation algorithms

- Progressive Reliability Transform (PRT), producing $M_1, M_2, M_3, \ldots , M_v, \ldots , M_{\text{max}}$

- The output list size $l_v$ is determined by
  $$l_v = \left\lfloor \frac{\Delta_{1,k-1}(C(M_v))}{k-1} \right\rfloor$$

- $\Delta_{1,k-1}C(M_v) = \deg_{1,k-1}(x^a y^b \mid \text{ord}(x^a y^b) = C(M_v))$

---

**Algorithm Reliability transform with stopping criterion $l_v$**

**Input:** Reliability matrix $\Pi$, $\Pi^*_{v-1}$, and the maximal output list size $l_v$ and multiplicity matrix $M_{v-1}$.

**Output:** Multiplicity matrix $M_v$.

**step 1:** Initiate $\Pi_v^* = \Pi^*_{v-1}$, $M_v = M_{v-1}$;

**step 2:** Find the largest entry $\pi^*_{ij}$ in $\Pi^*_v$ with the position $(i, j)$;

**step 3:** Update $\pi^*_{ij} = \frac{\pi_{ij}}{m_{ij}+2}$;

**step 4:** Update $m_{ij} = m_{ij} + 1$;

**step 5:** Compute $C(M_v) = \frac{1}{2} \sum_{i=0}^{q} \sum_{j=0}^{n} m_{ij}(m_{ij} + 1)$;

**step 6:** Compute $l_v^* = \left\lfloor \frac{\Delta_{1,k-1}(C(M_v))}{k-1} \right\rfloor$

**step 7:** If $l_v^* > l_v$, return $M_v$; otherwise go to step 2.
II. Implementation algorithms

- Progressive Interpolation (PIP)

From iteration $v \to v + 1$:
1) Generate an incremental polynomial group
   $$\Delta G_v = \{y^{l_v+1}, y^{l_v+2}, \ldots, y^{l_{v+1}}\}$$
   Perform $PIP(\Lambda(M_v), \Delta G_v) \to \Delta G_v'$, then update the new polynomial group as
   $$G_{v+1} = G_v \cup \Delta G_v'$$
2) For the updated polynomial group $G_{v+1}$, perform $PIP(\Lambda(\Delta M_{v+1}), G_{v+1}) \to G_{v+1}'$. 

- From iteration $v \to v + 1$: 
  1) Generate an incremental polynomial group
     $$\Delta G_v = \{y^{l_v+1}, y^{l_v+2}, \ldots, y^{l_{v+1}}\}$$
     Perform $PIP(\Lambda(M_v), \Delta G_v) \to \Delta G_v'$, then update the new polynomial group as
     $$G_{v+1} = G_v \cup \Delta G_v'$$
     2) For the updated polynomial group $G_{v+1}$, perform $PIP(\Lambda(\Delta M_{v+1}), G_{v+1}) \to G_{v+1}'$. 

II. Complexity reduction

- Computational complexity ($O$): the averaged number of finite field arithmetic operations for decoding one codeword frame;
- Complexity reduction ($\Theta$):

$$\Theta = \frac{O_{ASD} - O_{PASD}}{O_{ASD}} \times 100\%$$

- The (15, 5) RS code
II. Complexity reduction

- Measurement of the decoding parameter $l$

Measure the assignment of $l$ with respect to the channel quality for (15,5) RS code

<table>
<thead>
<tr>
<th>$SNR$</th>
<th>$l$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2dB</td>
<td></td>
<td>21.2130</td>
<td>15.8959</td>
<td>10.2188</td>
<td>7.0340</td>
<td>5.2340</td>
<td>4.0986</td>
<td>2.6862</td>
<td>2.6031</td>
<td>1.7170</td>
<td>29.2994</td>
</tr>
<tr>
<td>5dB</td>
<td></td>
<td>81.0490</td>
<td>12.7920</td>
<td>3.2638</td>
<td>1.0861</td>
<td>0.5532</td>
<td>0.3028</td>
<td>0.1745</td>
<td>0.1230</td>
<td>0.1048</td>
<td>5.5078</td>
</tr>
<tr>
<td>8dB</td>
<td></td>
<td>99.9339</td>
<td>0.0638</td>
<td>0.0014</td>
<td>0.0004</td>
<td>0.0003</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
II. Performance evaluation

- The (15, 5) RS code with BPSK, over AWGN channel
II. Performance evaluation

- Successful decoding criterion: \( S_M(\vec{C}) > \text{deg}_{1,k-1}(Q(x, y)) \)

- Conventional ASD algorithm might ‘overkill’ the decoding problem

- Example: performing ASD and PASD with \( l = 10 \)

<table>
<thead>
<tr>
<th>( l )</th>
<th>( S_M(C) )</th>
<th>( \text{deg}_{1,k-1}(Q(x, y)) )</th>
<th>( S_M(C) )</th>
<th>( \text{deg}_{1,k-1}(Q(x, y)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>&lt; 8</td>
<td>4</td>
<td>&lt; 8</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>&lt; 12</td>
<td>10</td>
<td>&lt; 12</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>&lt; 16</td>
<td>13</td>
<td>&lt; 16</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>&lt; 20</td>
<td>19</td>
<td>&lt; 20</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>&lt; 24</td>
<td>21</td>
<td>&lt; 24</td>
</tr>
<tr>
<td>6</td>
<td>27</td>
<td>&lt; 28</td>
<td>27</td>
<td>&lt; 28</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>&lt; 32</td>
<td>30</td>
<td>&lt; 32</td>
</tr>
<tr>
<td>8</td>
<td>34</td>
<td>&lt; 36</td>
<td>34</td>
<td>&lt; 36</td>
</tr>
<tr>
<td>9</td>
<td>41</td>
<td>&gt; 40</td>
<td>41</td>
<td>&gt; 40</td>
</tr>
<tr>
<td>10</td>
<td>44</td>
<td>= 44</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

An example based on (15,5) RS code for understanding why the PASD algorithm can outperform the ASD algorithm.
Conclusions

- Construction of a Hermitian code and some of its properties;
- Hermitian code can be a promising candidate to replace RS code in future applications;
- Algebraic soft-decoding of Hermitian codes, including the interpolated zero condition, validity of the decoding, optimal performance bound and complexity reduction approaches;
- Modernised algebraic soft decoding algorithm: a progressive approach;
- Two key steps of PASD: progressive reliability transform & progressive interpolation;
- Optimises both decoding complexity and performance;
- A general approach for all sorts of algebraic decoding problems.
Future work

- A continue thinking:
  PASD algorithm → performance ~ $\Pi$ dependent;
  → complexity ~ $\Pi$ dependent;

- An priori process to the PASD algorithm can be introduced to enhance the reliability of $\Pi$, enabling both a performance improvement and a faster convergence of decoding complexity.


The UK government Overseas Research Scholarship (ORS) scheme, supporting my PhD engagement (Part I of the presentation).

The National Natural Science Foundation of China (NSFC), supporting the proposed work of Part II. Project: Advanced coding technology for future storage devices, ID: 61001094. Role: principle investigator (PI).

Siyun Tang for implementing the PASD algorithm and Prof. Xiao Ma for his thoughtful discussion.
Thank you!