PETERSON – GIORENSTEIN – ZIELER Decoder

for a RS code which can correct 't' errors. ie $n-k = 2t$

START

↓

Compute Syndromes
S_l ; $l = 1, 2, \dots 2t$

↓

$V = t$

↓

$$M = \begin{bmatrix} S_1 & S_2 & \dots & S_V \\ S_2 & & & \\ \vdots & & & \\ S_V & \dots & \dots & S_{2V-1} \end{bmatrix}$$

↓

is $\det(M) = 0$

YES → $V = V - 1$ → (back to M)

NO ↓

$$\begin{bmatrix} \Lambda_V \\ \Lambda_{V-1} \\ \vdots \\ \Lambda_V \end{bmatrix} = M^{-1} \begin{bmatrix} S_{V+1} \\ \vdots \\ \vdots \\ S_{2V} \end{bmatrix}$$

Efficient implementation ⟸ MASEY'S Algorithm

"finding coefficients of error locating polynomial"

↓

find roots of $1 + \Lambda_1 x + \Lambda_2 x^2 \dots \Lambda_V x^V$
let roots be X_l ; $l = 1, 2 \dots V$

Chien Search ⟸

↓

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_V \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & \dots & X_V \\ X_1^2 & X_2^2 & \dots & X_V^2 \\ \vdots & & & \\ X_1^V & X_2^V & \dots & X_V^V \end{bmatrix}^{-1} \begin{bmatrix} S_1 \\ \vdots \\ S_V \end{bmatrix}$$

Efficient implementation ⟸ FORNEY'S Algorithm

"finding Error magnitudes"

↓

STOP

error polynomial $e(x) = Y_1 X_1 + Y_2 X_2 \cdots + Y_v X_v$.

eg: two error case

$X_1 = \alpha^3$ and $Y_1 = \alpha^4$

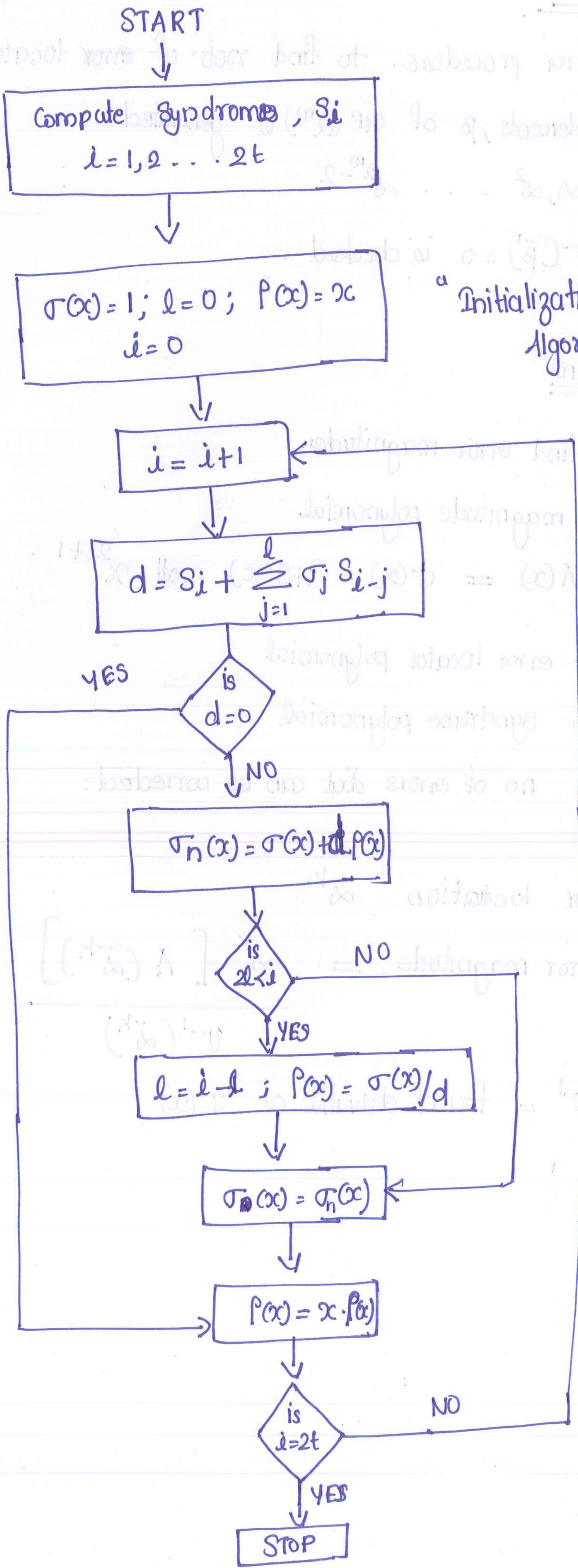
$X_2 = \alpha^7$ $Y_2 = \alpha$

$$e(x) = \alpha^4 x^3 + \alpha x^7$$

Corrected polynomial = recieved polynomial + $e(x)$.

MASEY's Algorithm

To find error locator polynomial $\sigma(x)$.



"Initialization of Algorithm"

Chien search

- Trial and Error procedure to find roots of error locator polynomial.
- All non zero elements, β of $GF(2^m)$ is generated

$$1, \alpha, \alpha^2 \ldots \alpha^{2^m-2}$$

- condition $\sigma(\beta^{-1}) = 0$ is checked.

forney's Algorithm

- used to find error magnitudes.
- find error magnitude polynomial

$$\Lambda(x) = \sigma(x) \cdot (1+S(x)) \bmod x^{2t+1}.$$

- $\sigma(x)$ = error locator polynomial

  $S(x)$ = syndrome polynomial.

  t = no of errors that can be corrected.

- for error location α^k

$$\text{error magnitude} = \frac{\alpha^k \left[\Lambda(\alpha^{-k})\right]}{\sigma'(\alpha^{-k})}$$

σ' = formal derivative of $\sigma(x)$