STANFORD

Supplementary Lecture 8A

Equivalent Encoders

January 29, 2026

JOHN M. CIOFFI

Hitachi Professor Emeritus (recalled) of Engineering

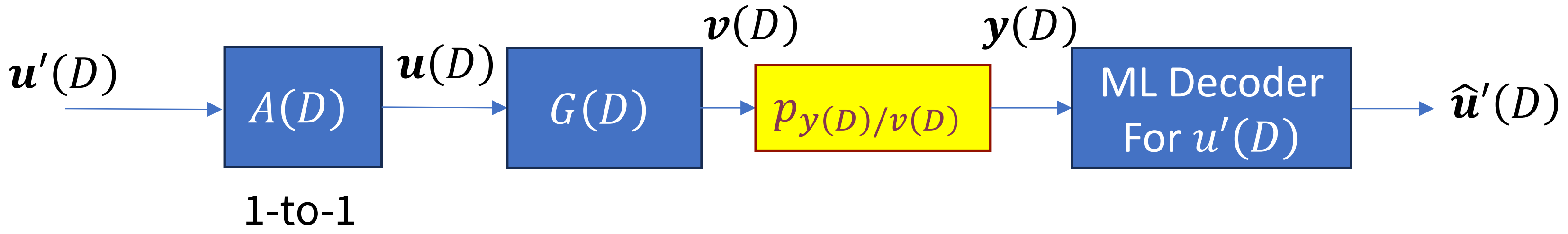
Instructor EE379A – Winter 2026

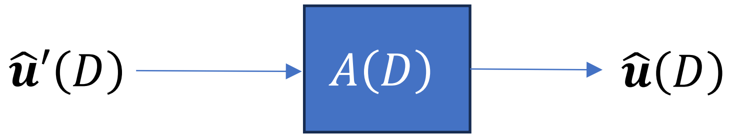
Announcements & Agenda

- Announcements
  - This is in Appendix B with worked examples

- Today
  - Encoder's Mapping
  - Invariant Factors Decomposition & minimal realization
  - Example

Decoders' Mappings





The codewords $\{\boldsymbol{v}(D)\}$ Are the SAME

- The invertible $A(D)$ can sometimes simplify the ML Decoder.
- The detection of $\boldsymbol{u}'(D)$ has same P_e as detection of $\boldsymbol{u}(D)$
- The equivalent encoder $A(D) \cdot G(D)$ might:
  - have fewer states, so the Viterbi/BCJR/SOVA then is simpler,
  - be systematic for systems that only want parity bits to differ from input bits, or
  - have software/hardware implementation that is already available.

Invariant Factors Decomposition

Appendix B.4

Invariant Factors Decomposition

- Any polynomial encoder $G(D) = A(D) \cdot \Gamma(D) \cdot B(D)$ where
  - $A(D)$ and $B(D)$ are both in $F[D]$, so all entries are FIR, unimodular $|A| = |B| = 1$.
  - $\Gamma(D)$ is diagonal (on left) and has all elements in $F_r(D)$
    - so IIR with all positive or 0 powers of D in numerator and denominator, which is also monic.

$$\boxed{A(D) \cdot [\Gamma(D) \quad \mathbf{0}] \cdot \begin{bmatrix} \boldsymbol{G_{basic}(D)} \\ don't\ care \end{bmatrix}}$$

- This is also Smith-Normal Form, BUT in a finite field.
  - Matlab does NOT have a program that does this.
  - I tried to get ChatGPT to create one, but it was too hard for ChatGPT, which kept making errors.
  - Thus, best I know is the process, with several examples I worked by hand in Appendix B.
    - If a student wants to attempt this in matlab program – lots of extra credit.

- The IFD factorization is tedious, but straightforward. Appendix B.4 has examples.
  - If the original encoder has feedback, the IVD clears the denominators by multiplying by LCM, then divides again after doing the factorization. Thus, the IVD only needs to work on FIR matrices.

- This is one of 2 steps in ensuring a code uses minimal flip flops (smallest number of states) in its encoder. Since A and B have inverses and are 1-to-1, the invariant factors and top k rows of $B(D)$ determine all the codewords. Different encoders have different choices of unimodular A, B.

Minimal Encoder – 2nd step

- $G_{basic}(D)$ is feedback free (and has feedback-free inverse – first k columns of $B^{-1}(D)$)

- This "basic encoder" may not yet have minimum number of flip flops (smallest number of states).

- Appendix B lists a second process that forces iteratively the encoder to preserve both delay and degree (plus ν) of for any input sequence.
  - It's again a series of unimodular $A(D) - like$ operations on the rows of $G_{basic}(D)$ that do not change the code (the codewords are the same)

An Example follows:

Parity Code Tables, Feedback, and poly2trellis

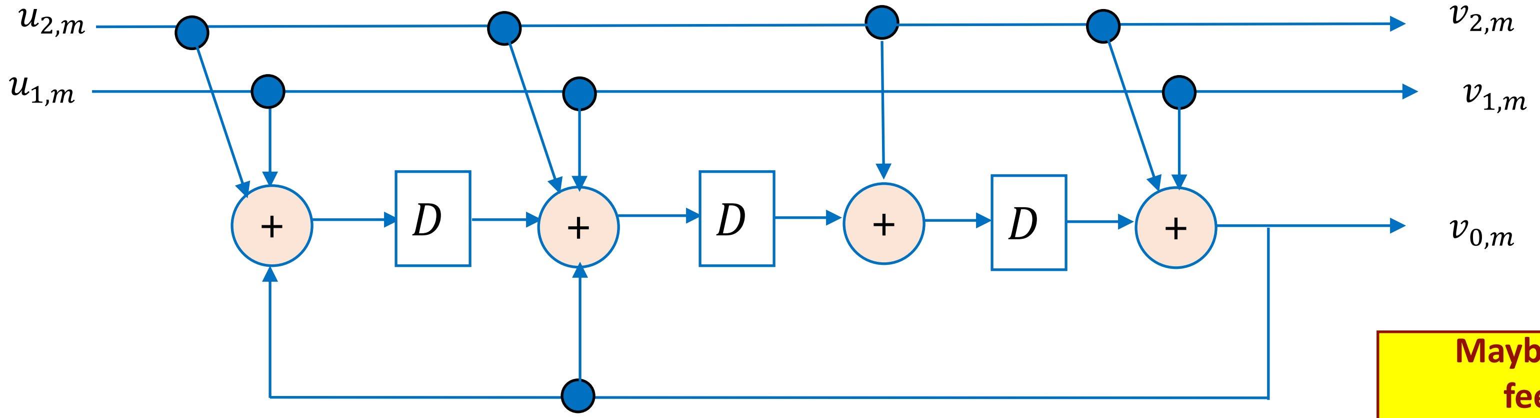
- Poly2trellis has a third input that is feedback – example best 8-state $r = 2/3$ conv code from tables

$$H(D) = [17 \quad 15 \quad 13] = [D^3 + D^2 + D + 1 \quad D^3 + D^2 + 1 \quad D^3 + D + 1]$$

$$H_{sys}(D) = \left[\frac{D^3 + D^2 + D^1 + 1}{D^3 + D + 1} \quad \frac{D^3 + D^2 + 1}{D^3 + D + 1} \quad 1\right]$$

$$G_{sys}(D) = \begin{bmatrix} 1 & 0 & \dfrac{D^3 + D^2 + D^1 + 1}{D^3 + D + 1} \\ 0 & 1 & \dfrac{D^3 + D^2 + 1}{D^3 + D + 1} \end{bmatrix}$$

- Circuit has 8 states (3 flip flops)



Maybe we want feedback, or maybe not

What does Matlab do with this 2/3 code?

```
>> tfeed=poly2trellis([4 4],[13 0 17 ; 0 13 15], [13 13])

tfeed =

     numInputSymbols: 4
    numOutputSymbols: 8
           numStates: 64  OUCH!
          nextStates: [64 × 4 double]
             outputs: [64 × 4 double]
```

- I could find no way to use this command other than the above valid (but nonminimal trellis).
- The matlab page examples do the same thing – increase number of states excessively.
- This is NOT a problem if code is $r = {}^{1}/_{n}$, then number of states is preserved.
- Here it was square of number of states (64), for rate ¾, it would cube number of states.

But IFD provides a fix!

Work-Around uses IFD & minimal encoder

- See Appendix B that details step-by-step the IFD for this rate 2/3 systematic encoder
  - This produces:

$$G_{sys}(D) = \underbrace{\begin{bmatrix} 1+D+D^2+D^3 & 1+D+D^2 \\ 1+D^2+D^3 & D+D^2 \end{bmatrix}}_{\substack{A \\ |A|=1}} \cdot \underbrace{\begin{bmatrix} \frac{1}{D^3+D+1} & 0 \\ 0 & 1 \end{bmatrix}}_{\substack{\Gamma \\ |\Gamma|\neq 0}} \cdot \underbrace{\begin{bmatrix} D & 1+D^2 & 1+D^2 \\ 1+D & D & 1 \end{bmatrix}}_{G_{min}(D)}$$

- The first two matrices are 1-to-1, so only remap all possible binary inputs to the SAME codewords.
  - They do not affect the set of codewords (or the code).
- Minimal 8-state feedback-free encoder is $G_{min}(D) = \begin{bmatrix} D & 1+D^2 & 1+D^2 \\ 1+D & D & 1 \end{bmatrix}$.

- Encode with $G_{sys}(D)$ convenc.m has no issues (even though it uses 64 states) or just encode with 8 state circuit on L8:27; the codewords are the same (so MLSD will find closest codeword).
- Decoder assumes $G_{min}(D)$ and finds $\hat{u}_{min}(D)$; then $\hat{v}_{min}(D)$= $\hat{u}_{min}(D) \cdot G_{min}(D)$ - re-encodes the decoded.
- $\hat{u}_{sys}(D) = [\hat{v}_{2,min}(D) \quad \hat{v}_{1,min}(D)]$ because the original encoder was systematic – ignore $\hat{v}_{0,min}(D)$.
  - Further, any finite number of output errors only cause a finite (possibly less, but not more) number of input bit errors.

Example: 8-state rate 2/3 code

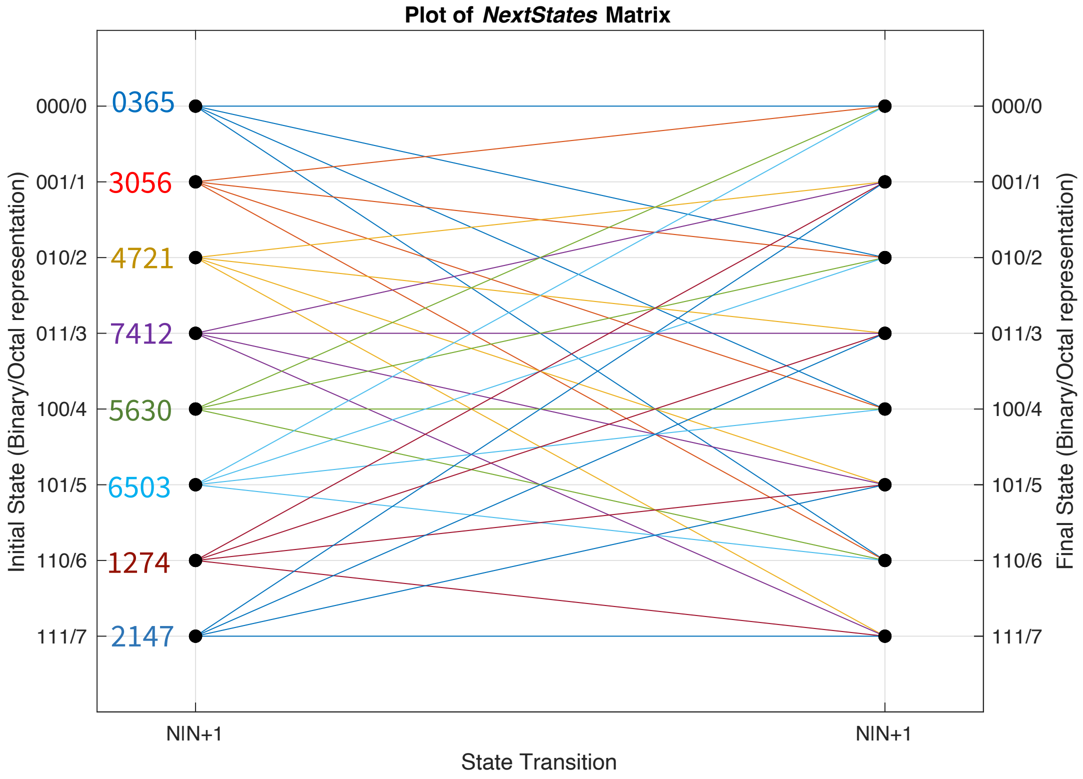
- Saving commands

```
tmin=poly2trellis([3 2], [2 5 5; 3 2 1])

    numInputSymbols: 4
   numOutputSymbols: 8
          numStates: 8
         nextStates: [8×4 double]
            outputs: [8×4 double]

>> tmin.nextStates
    0    4    2    6
    0    4    2    6
    1    5    3    7
    1    5    3    7
    0    4    2    6
    0    4    2    6
    1    5    3    7
    1    5    3    7

>> tmin.outputs
    0    6    3    5
    3    5    0    6
    4    2    7    1
    7    1    4    2
    5    3    6    0
    6    0    5    3
    1    7    2    4
    2    4    1    7
>> outmin=convenc([ 0 0  0 0  0 0  1 0  1 1  0 1  0 0  0 1],tmin)
    000    000    000    011  001    100    110    110
>> plotnextstates(tmin.nextStates)
```



Plot of *NextStates* Matrix
Initial State (Binary/Octal representation)
Final State (Binary/Octal representation)
000/0
001/1
010/2
011/3
100/4
101/5
110/6
111/7
0365
3056
4721
7412
5630
6503
1274
2147
NIN+1
NIN+1
State Transition

8-state decode

- Minimal Direct Works – dfree = 6

```
>> vitdec(outmin,tmin,6,'trunc','hard')
  00    00    00    10    11    01  00    01
>> inmin=
  00    00    00    10    11    01    00    01];
---------------
error2 = [ 0 0 1  0 0 0   0 0 0   0 0 0  0 0 0   0 0 0   0 1 0  0 0 0 ];  % 2 errors introduced
>> vitdec(+xor(outmin,error2),tmin,6,'trunc','hard')

  00    00    00    10    11    01    00    01
```

Have to leave spaces in matlab, but it looks better without them here

- Systematic feedback encoder – different output

```
>> tfeed=poly2trellis([4 4],[13 0 17 ; 0 13 15], [13 13])
    numInputSymbols: 4
   numOutputSymbols: 8
          numStates: 64
         nextStates: [64 × 4 double]
            outputs: [64 × 4 double]
>> outfeed=convenc([ 0 0     0 0     0 0     1 0     1 1     0 1    0 0     0 1],tfeed)
                              0 0 0   0 0 0   0 0 0   1 0 1   1 1 1   0 1 1  0 0 1  0 1 1 %systematic
>> informin=vitdec(outfeed,tmin,6,'trunc','hard')
                               00      00      00      11      01      11     00      00 % map differs
>> vmin = convenc(informin,tmin)  =
                               000     000     000    101     111     011    001     011
```

```
>> informin2=vitdec(+xor(outfeed,error2),tmin,6,'trunc','hard')
   0 0    0 0    0 0    1 1    0 1    1 1    0 1    1 1
>> vmin2 = convenc(informin2,tmin)
   000   000   000   101   111   011   001   011

>> outfeed  % check
   000  000  000  101  111  011  001  011

% So, this fixes matlab's high-complexity-trellis problem with 8-state decoder
```

This works for any decoder,
But, of course, most helpful
With matlab poly2trellis issues

STANFORD

End Lecture 8

See S8a for Invariant Factors Theorem