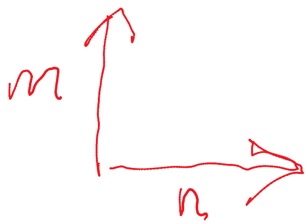
Modulation and Demodulation

Channel sharing

- Suppose we have TWO CARRIERS that are orthogongal to one another…then we can separate the effects of these two carrriers…
- Whoa….

Vectors and modulation



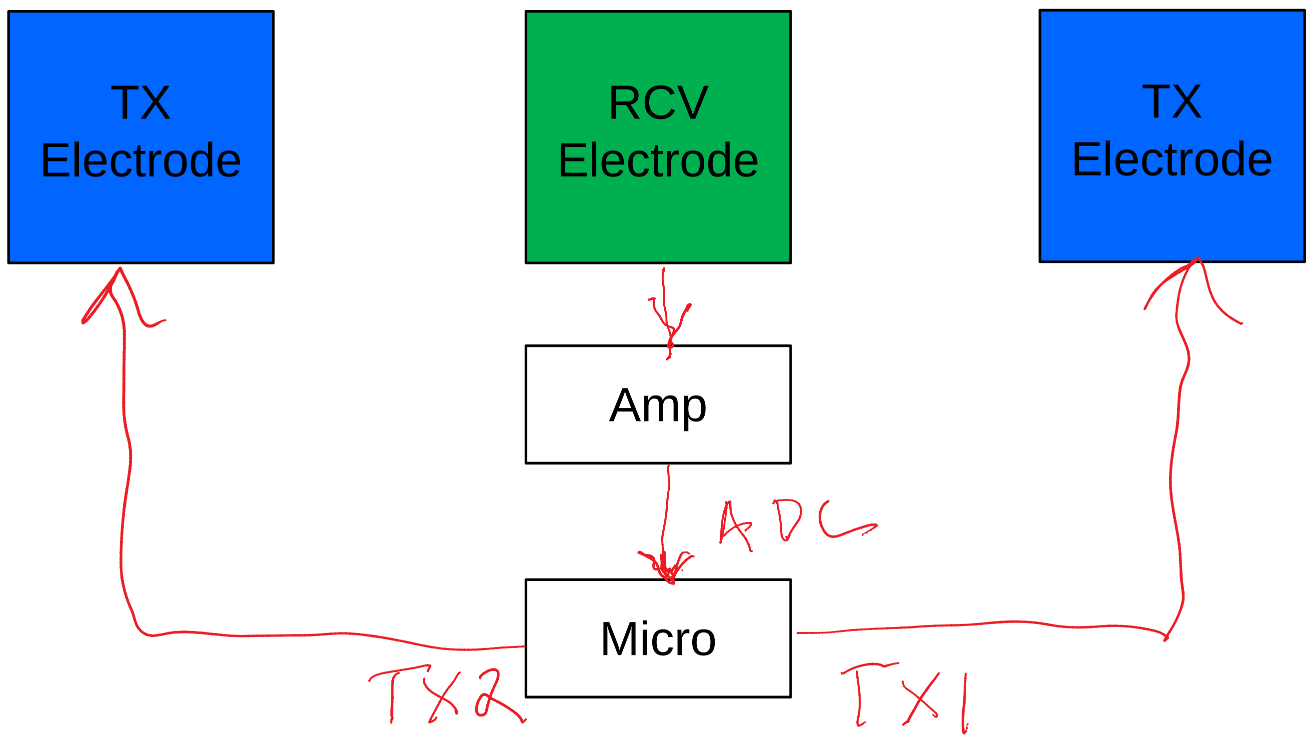
Vectors: **bold blue**
Scalars: not

S'pose m and n are orthogonal unit vectors.
Then inner products (dot products) are
<**m**,**m**>=1 <**n**,**n**>=1
<**m**,**n**>=<**n**,**m**>=0

Can interpret inner product as projection of vector 1 ("**v1**") onto vector 2 ("**v2**")…in other words, inner product of **v1** and **v2** tells us "how much of vector 1 is there in the direction of vector 2."

If a channel lets me send 2 orthogonal vectors through it, then
I can send two independent messages. Say I need to send two numbers, a and b…I can send a**m**+b**n** through the channel.
At the receive side I get a**m**+b**n**
Now I project onto m and onto n to get back the numbers:
<a**m**+b**n**, **m**>=<a**m**,**m**> + <b**n**, **m**>=a+0=a
<a**m**+b**n**, **n**>=<a**m**,**n**> + <b**n**, **n**>=0+b=b
The initial multiplication is modulation; the projection to separate the signals is demodulation. Each channel sharing scheme←→a set of basis vectors.
In single-channel e-field sensing, the "carrier" we transmit is **m**, the sensed value is a, and the noise is **n**

Physical set up for multiplexed sensing



We can measure multiple sense channels simultaneously, sharing 1 RCV electrode, amp, and ADC!

Choice of TX wave forms determines multiplexing method:

- TDMA --- Time division: TXs take turns
- FDMA --- Frequency division: TXs use different frequencies
- CDMA ---- Code division: TXs use different coded waveforms

In all cases, what makes it work is ~orthogonality of the TX waveforms!

Single channel sensing / communication

$acc = \langle C, ADC \rangle$

Where C is the carrier vector and ADC is the vector of samples.
Let's write out ADC:

$$ADC = hC$$

Where h (hand) is sensed value and hC means scalar h x vector C

Acc
$= \langle C, hC \rangle$
$= h \langle C, C \rangle$
$= h$
if $\langle C, C \rangle = 1$

Multi-access sensing / communication

Suppose we have two carriers, C^1 and C^2

And suppose they are orthogonal, so that $< C^1, C^2 >=0$

The received signal is

$ADC = h^1C^1+h^2C^2$

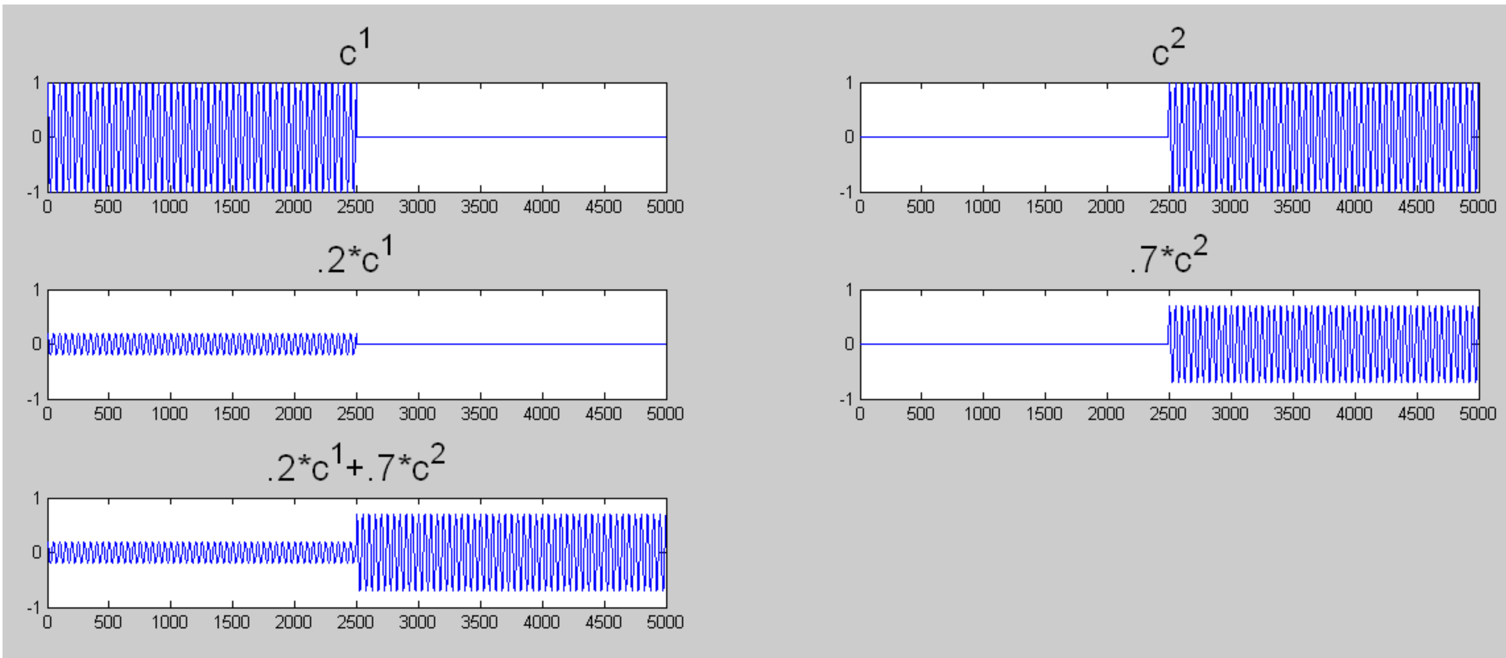
Let's demodulate with C^1:

$$
\begin{aligned}
&acc \\
&=<C^1, ADC > \\
&=< C^1, h^1C^1+h^2C^2 > \\
&=< C^1, h^1C^1> + <C^1,h^2C^2 > \\
&=h^1< C^1, C^1> + h^2<C^1,C^2 > \\
&= h^1
\end{aligned}
$$

If $< C^1, C^1> = 1$ *and* $< C^1, C^2> = 0$

TDMA

Abstract view



Verify that $\langle C^1, C^2 \rangle = 0$

Modulated carriers

Sum of modulated carriers

$\langle C^1, .2C^1 + .7C^2 \rangle =$

$\langle C^1, .2C^1 \rangle + \langle C^1, .7C^2 \rangle =$

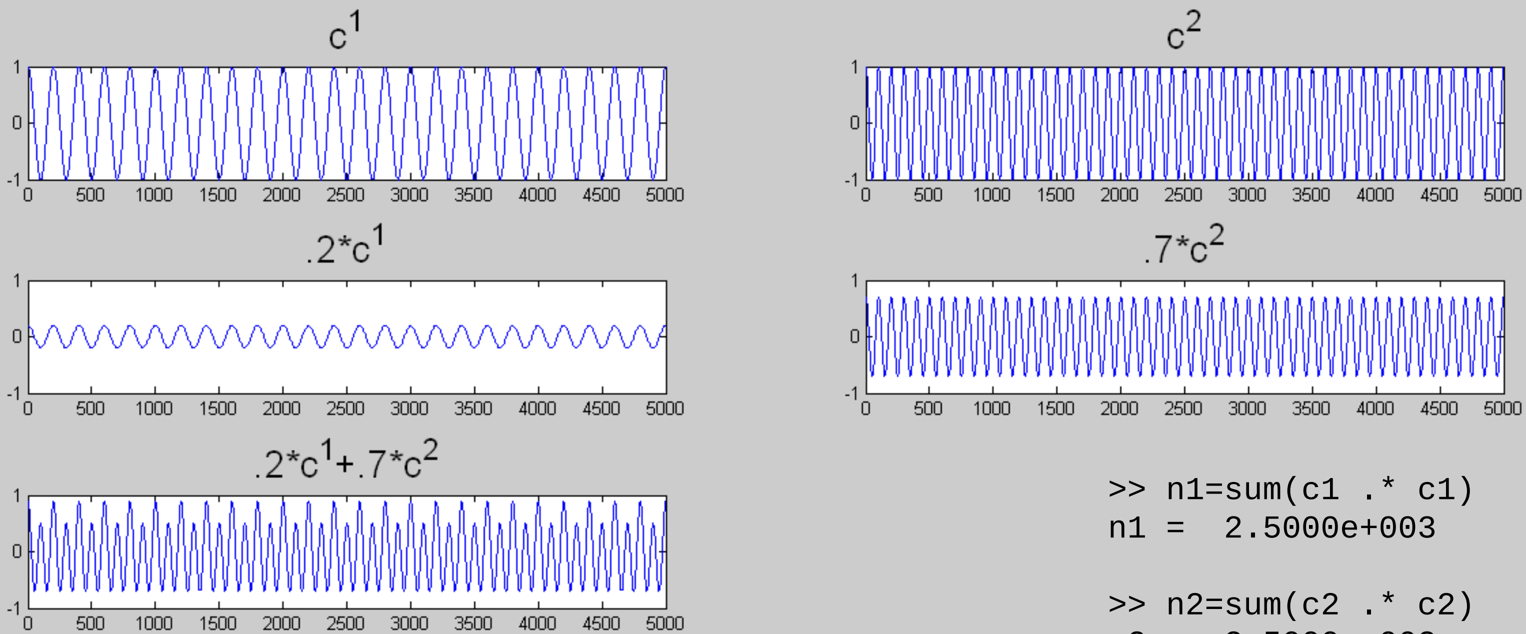
$.2 \langle C^1, C^1 \rangle + 0$

Horizontal axis: time

Vertical axis: amplitude (arbitrary units)

FDMA

Abstract view

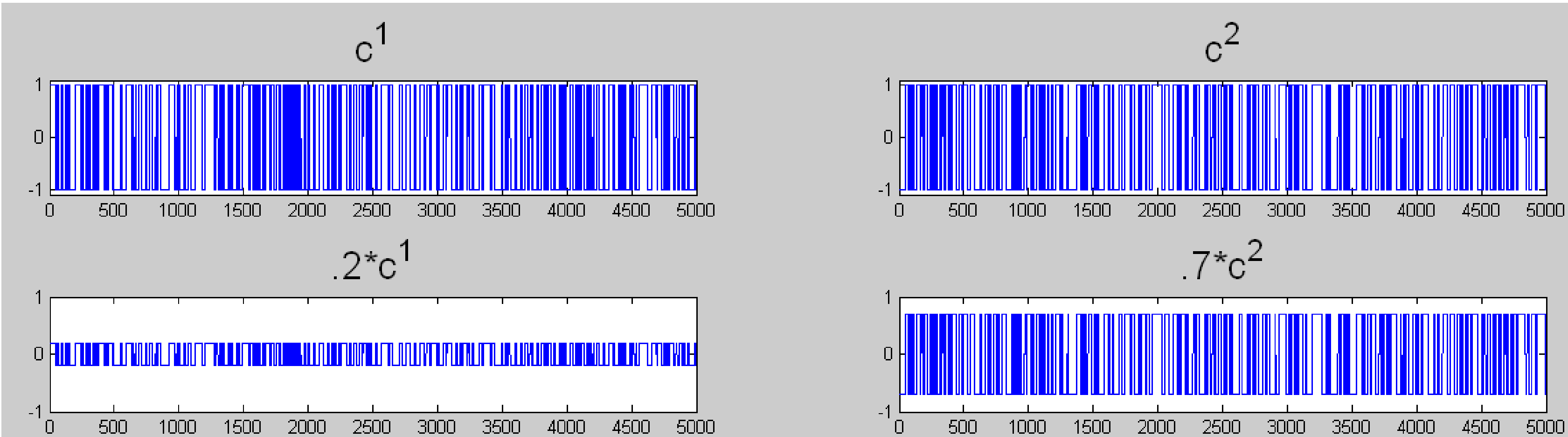


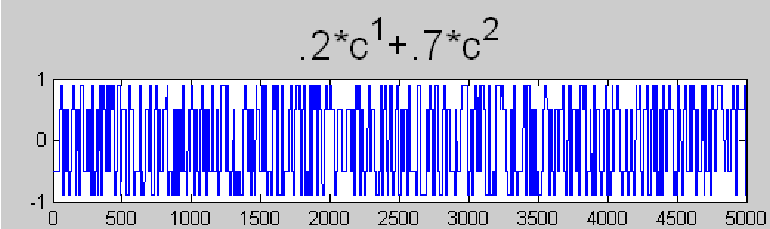

```
>> n1=sum(c1 .* c1)
n1 =  2.5000e+003

>> n2=sum(c2 .* c2)
n2 =  2.5000e+003

>> n12=sum(c1 .* c2)
n12 = -8.3900e-013

>> rcv = .2*c1 + .7*c2;
>> sum(c1/n1 .* rcv)
ans =    0.2000

>> sum(c2/n2 .* rcv)
ans =    0.7000
```

Horizontal axis: time
Vertical axis: amplitude (arbitrary units)

CDMA

S'pose we pick random carriers: `c1 = 2*(rand(1,500)>0.5)-1;`






```
>> n1=sum(c1 .* c1)
n1 =     5000

>> n2=sum(c2 .* c2)
n2 =     5000

>> n12=sum(c1 .* c2)
n12 =  -360

>> rcv = .2*c1 + .7*c2;
>> sum(c1/n1 .* rcv)
ans =  0.1496

>> sum(c2/n2 .* rcv)
ans =  0.6856
```

Horizontal axis: time
Vertical axis: amplitude (arbitrary units)

Note: Random carriers here consist of 500 rand values repeated 10 times each for better display

LFSRs (Linear Feedback Shift Registers)

The right way to generate pseudo-random carriers for CDMA

- A simple pseudo-random number generator
  - Pick a start state, iterate
- Maximum Length LFSR visits all states before repeating
  - Based on primitive polynomial…iterating LFSR equivalent to multiplying by generator for group
  - Can analytically compute auto-correlation
- This form of LFSR is easy to compute in HW (but not as nice in SW)
  - Extra credit: there is another form that is more efficient in SW
- Totally uniform auto-correlation

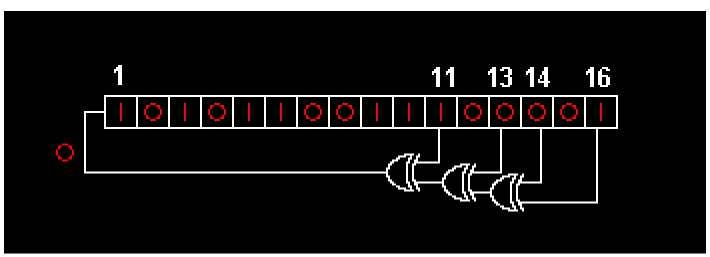


Image source: wikipedia

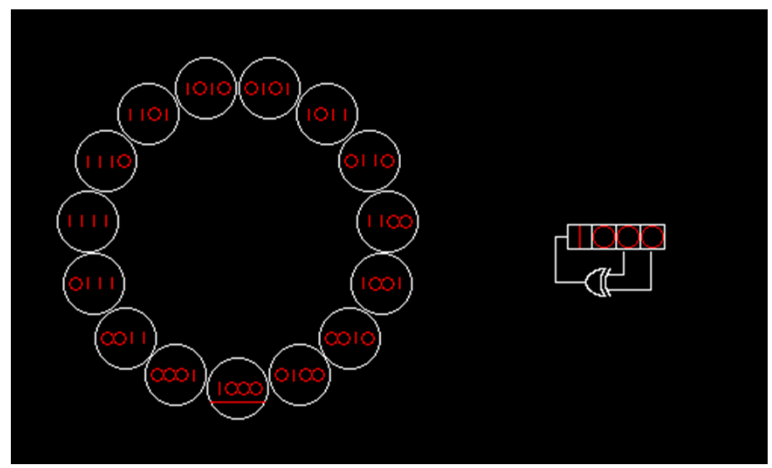


Image source: wikipedia

LFSR TX

8 bit LFSR with taps at 3,4,5,7 (counting from 0). Known to be maximal.

```
for (k=0;k<3;k++) { // k indexes the 4 LFSRs
    low=0;
    if(lfsr[k]&8) // tap at bit 3
      low++; // each addition performs XOR on low bit of low
    if(lfsr[k]&16) // tap at bit 4
      low++;
    if(lfsr[k]&32) // tap at bit 5
      low++;
    if(lfsr[k]&128) // tap at bit 7
      low++;
    low&=1; // keep only the low bit
    lfsr[k]<<=1;  // shift register up to make room for new bit
    lfsr[k]&=255; // only want to use 8 bits (or make sure lfsr is 8 bit var)
    lfsr[k]|=low; // OR new bit in
}
OUTPUT_BIT(TX0,lfsr[0]&1); // Transmit according to LFSR states
OUTPUT_BIT(TX1,lfsr[1]&1);
OUTPUT_BIT(TX2,lfsr[2]&1);
OUTPUT_BIT(TX3,lfsr[3]&1);
```

LFSR demodulation

```
meas=READ_ADC(); // get sample…same sample will be processed in different ways
for(k=0;k<3;k++) {
    if(lfsr[k]&1) // check LFSR state
      accum[k]+=meas;  // make sure accum is a 16 bit variable!
    else
      accum[k]-=meas;
}
```

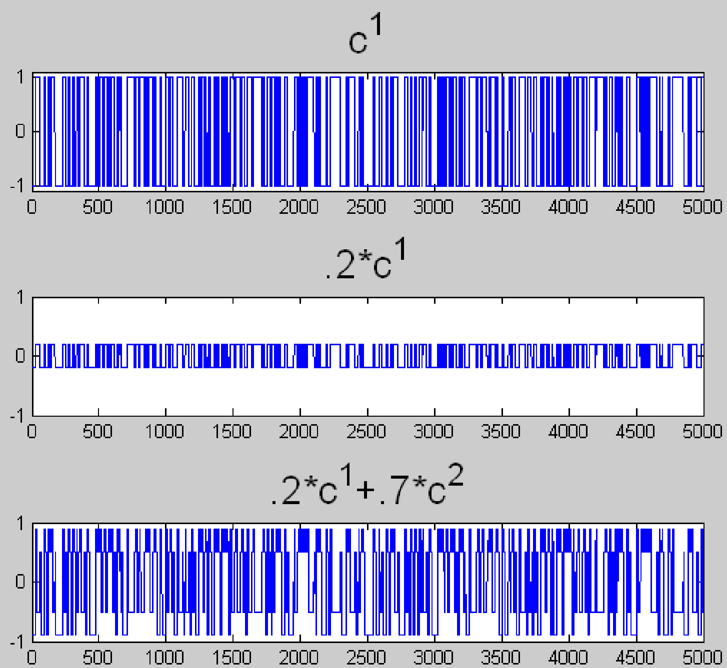
LFSR state sequence

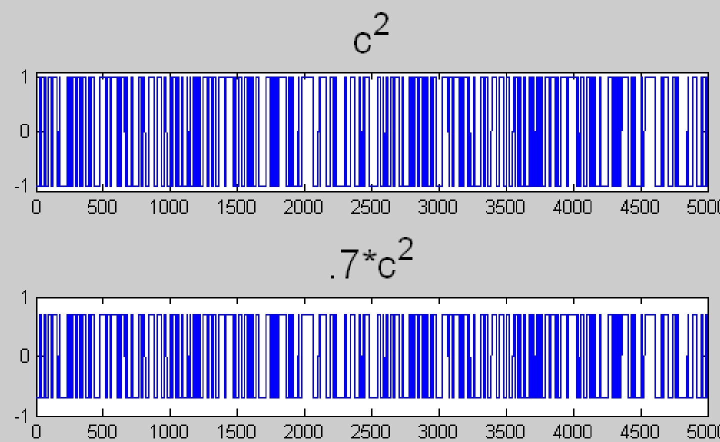
```
>> lfsr1(1:255)

ans =
2     4     8    17    35    71   142    28    56   113   226   196   137    18
37    75   151    46    92   184   112   224   192   129     3     6    12    25
50   100   201   146    36    73   147    38    77   155    55   110   220   185
114   228   200   144    32    65   130     5    10    21    43    86   173    91
182   109   218   181   107   214   172    89   178   101   203   150    44    88
176    97   195   135    15    31    62   125   251   246   237   219   183   111
222   189   122   245   235   215   174    93   186   116   232   209   162    68
136    16    33    67   134    13    27    54   108   216   177    99   199   143
30    60   121   243   231   206   156    57   115   230   204   152    49    98
197   139    22    45    90   180   105   210   164    72   145    34    69   138
20    41    82   165    74   149    42    84   169    83   167    78   157    59
119   238   221   187   118   236   217   179   103   207   158    61   123   247
239   223   191   126   253   250   244   233   211   166    76   153    51   102
205   154    53   106   212   168    81   163    70   140    24    48    96   193
131     7    14    29    58   117   234   213   170    85   171    87   175    95
190   124   249   242   229   202   148    40    80   161    66   132     9    19
39    79   159    63   127   255   254   252   248   240   225   194   133    11
23    47    94   188   120   241   227   198   141    26    52   104   208   160
64   128     1
```

LFSR output

```
>> c1(1:255)  (EVEN LFSR STATE ➔ -1, ODD LFSR STATE➔ +1)

ans =
    -1    -1    -1     1     1     1    -1    -1    -1     1    -1    -1     1    -1
     1     1     1    -1    -1    -1    -1    -1    -1     1     1    -1    -1     1
    -1    -1     1    -1    -1     1     1    -1     1     1     1    -1    -1     1
    -1    -1    -1    -1    -1     1    -1     1    -1     1     1    -1     1     1
    -1     1    -1     1     1    -1    -1     1    -1     1     1    -1    -1    -1
    -1     1     1     1     1     1    -1     1     1    -1     1     1     1     1
    -1     1    -1     1     1     1    -1     1    -1    -1    -1     1    -1    -1
    -1    -1     1     1    -1     1     1    -1    -1    -1     1     1     1     1
    -1    -1     1     1     1    -1    -1     1     1    -1    -1    -1     1    -1
     1     1    -1     1    -1    -1     1    -1    -1    -1     1    -1     1    -1
    -1     1    -1     1    -1     1    -1    -1     1     1     1    -1     1     1
     1    -1     1     1    -1    -1     1     1     1     1    -1     1     1     1
     1     1     1    -1     1    -1    -1     1     1    -1    -1     1     1    -1
     1    -1     1    -1    -1    -1     1     1    -1    -1    -1    -1    -1     1
     1     1    -1     1    -1     1    -1     1    -1     1     1     1     1     1
    -1    -1     1    -1     1    -1    -1    -1    -1     1    -1    -1     1     1
     1     1     1     1     1     1    -1    -1    -1    -1     1    -1     1     1
     1     1    -1    -1    -1     1     1    -1     1    -1    -1    -1    -1    -1
    -1    -1     1
```

CDMA by LFSR

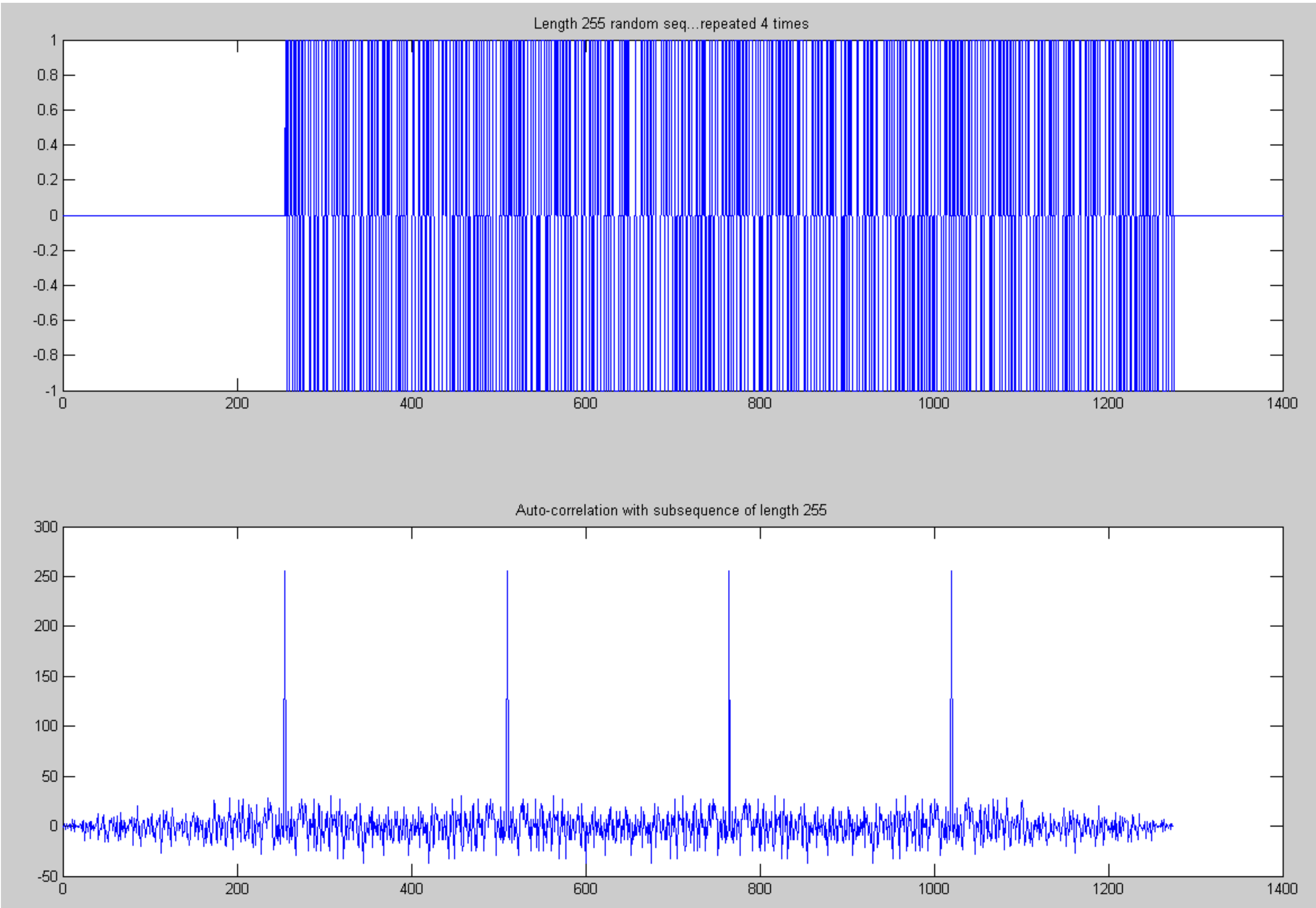





```
>> n1 = sum(c1.*c1)
n1 =    5000

>> n2 = sum(c2.*c2)
n2 =    5000

>> n12 = sum(c1.*c2)
n12 =    -60

>> rcv = .2 *c1 + .7*c2;
>> sum(c1/n1 .* rcv)
ans =    0.1916

>> sum(c2/n2 .* rcv)
ans =    0.6976
```

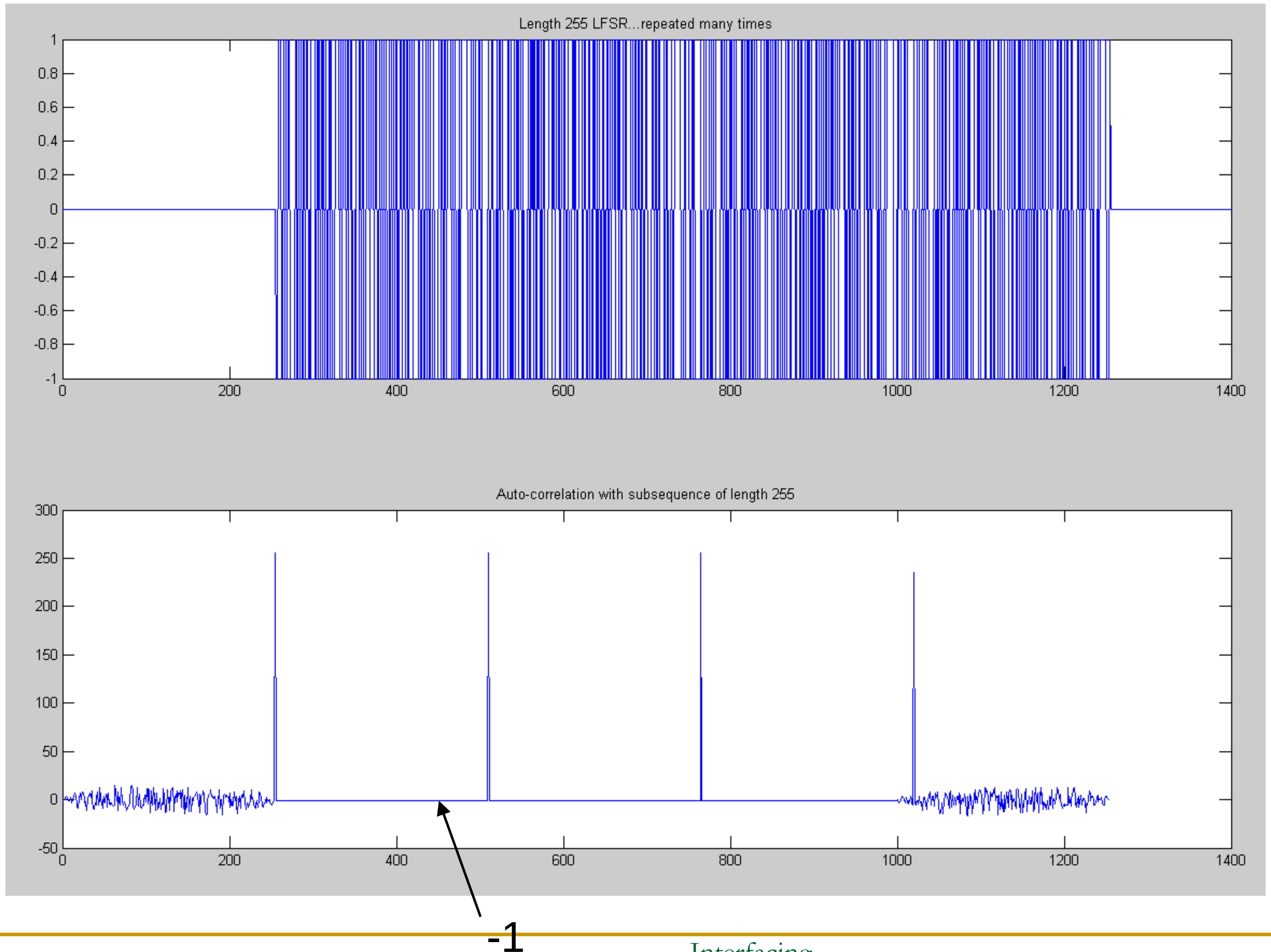
Note: CDMA carriers here consist of 500 pseudorandom values repeated 10 times each for better display

Autocorrelation of pseudo-random (non-LFSR) sequence of length 255



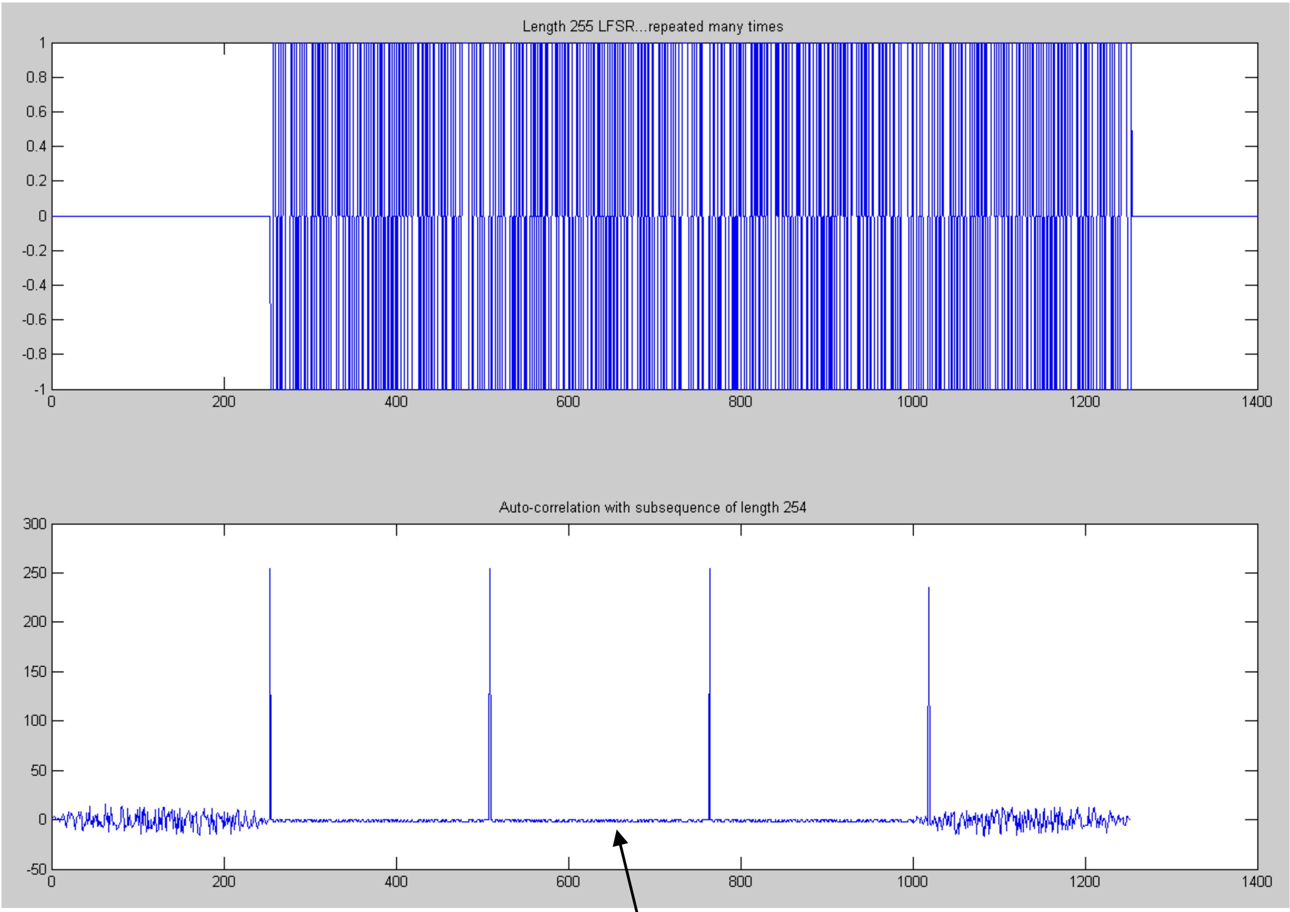
PR seq
Generated
w/ Matlab
rand cmd

Autocorrelation (full length 255 seq)



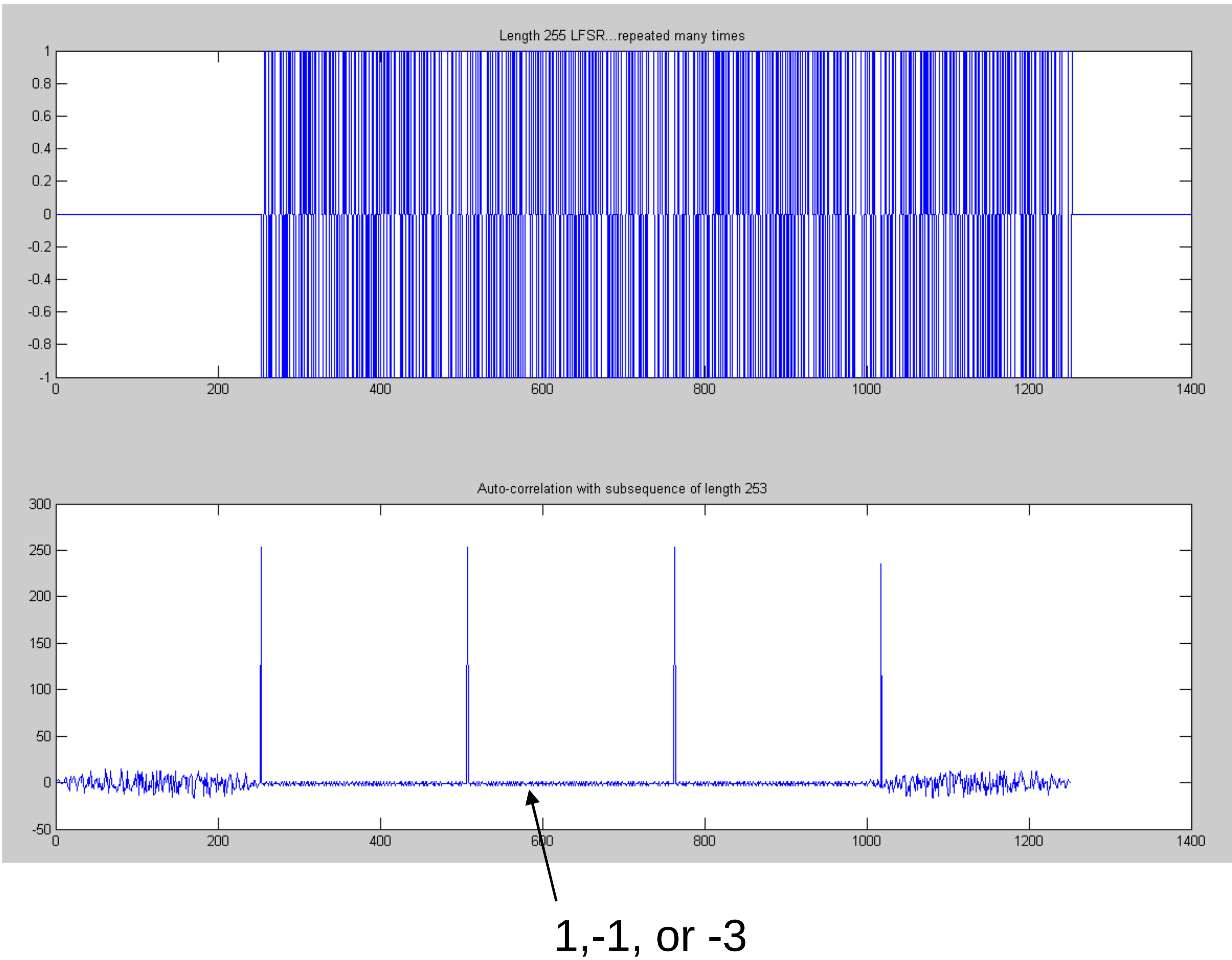
End of lecture

Autocorrelation (length 254 sub-seq)

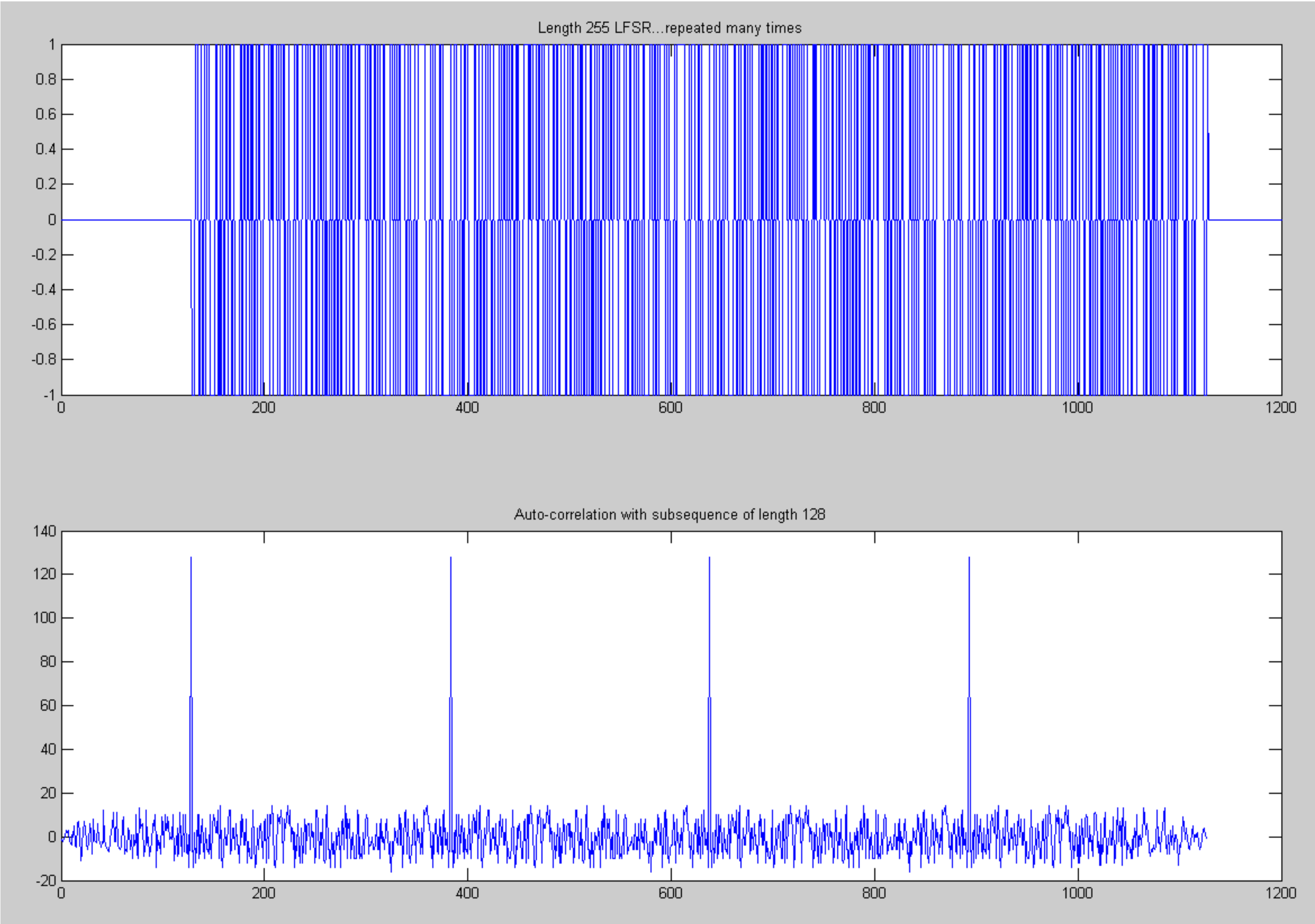


0 or -2

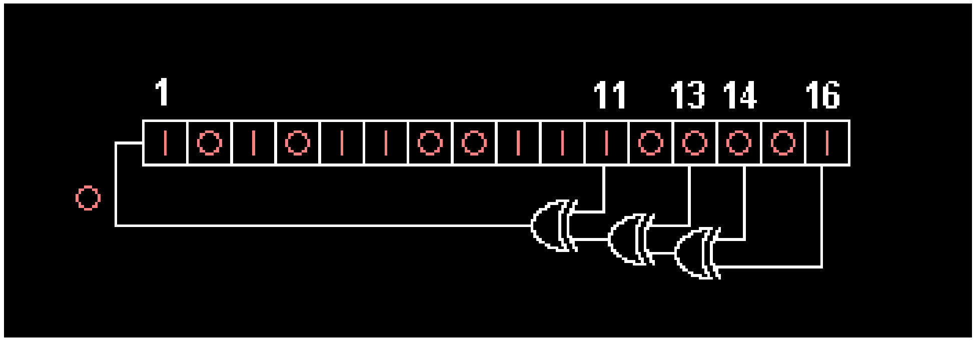
Autocorrelation (length 253 sub-seq)



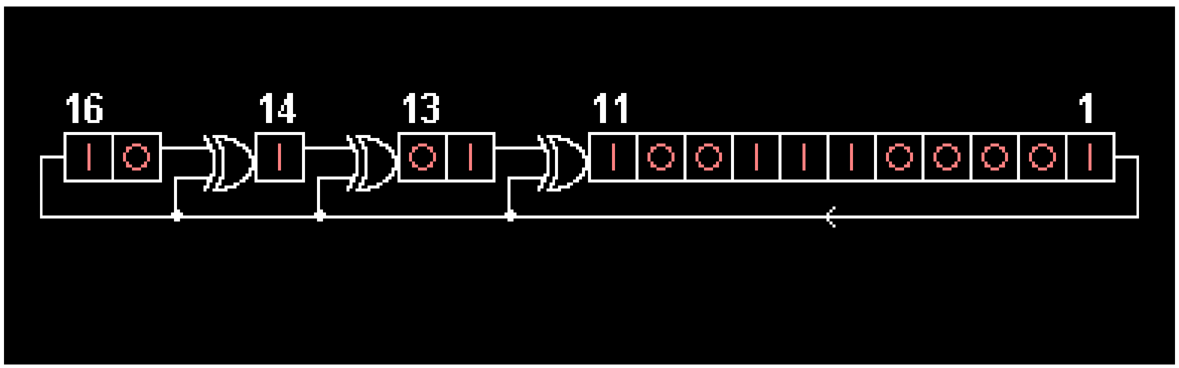
Autocorrelation (length 128 sub-seq)



LFSRs…one more thing…



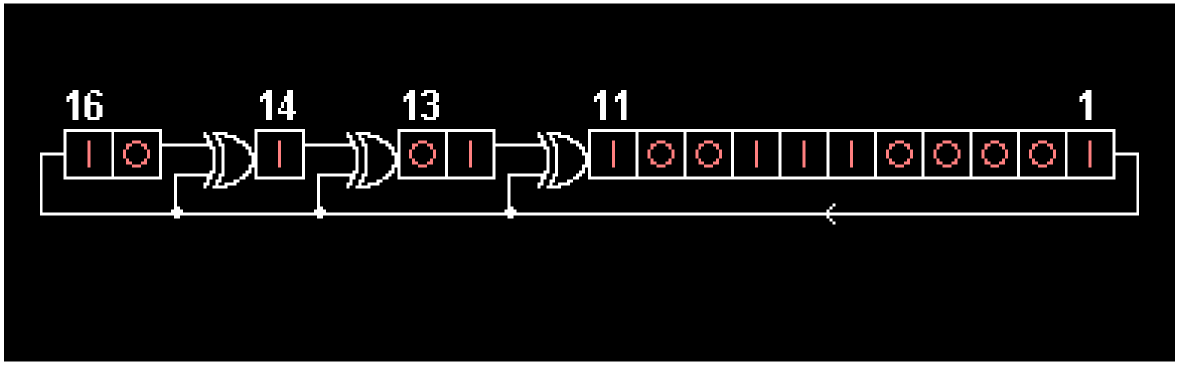
"Fibonacci"
"Standard"
"Many to one"
"External XOR"
LFSR



"Galois"
"One to many"
"Internal XOR"
LFSR
Faster in SW!!

Note: In a HW implementation, if you have XOR gates with as many inputs as you want, then the upper configuration is just as fast as the lower. If you only have 2 input XOR gates, then the lower implementation is faster in HW since the XORs can occur in parallel.

Advantage of Galois LFSR in SW



"Galois"
"Internal XOR"
"One to many"
LFSR

Faster in SW because XOR can happen word-wise (vs the multiple bit-wise tests that the Fibonacci configuration needs)

```
#include <stdint.h>
uint16_t lfsr = 0xACE1u;
unsigned int period = 0;
do {
  unsigned lsb = lfsr & 1; /* Get lsb (i.e., the output bit). */
  lfsr >>= 1; /* Shift register */
  if (lsb == 1) /* Only apply toggle mask if output bit is 1. */
    lfsr ^= 0xB400u; /* Apply toggle mask, value has 1 at bits corresponding
                      * to taps, 0 elsewhere. */
  ++period;
} while(lfsr != 0xACE1u);
```

LFSR in a single line of C code!

```
#include <stdint.h>
uint16_t lfsr = 0xACE1u;
unsigned period = 0;
do { /* taps: 16 14 13 11; char. poly: x^16+x^14+x^13+x^11+1 */
  lfsr = (lfsr >> 1) ^ (-(lfsr & 1u) & 0xB400u);
  ++period;
} while(lfsr != 0xACE1u);
```

NB: The minus above is two's complement negation…here the result is all zeros or all ones…that is ANDed that with the tap mask…this ends up doing the same job as the conditional from the previous implementation. Once the mask is ready, it is XORed to the LFSR

Some "polynomials" (tap sequences) for Max. Length LFSRs

Bits	Feedback polynomial	Period
n		$2^n - 1$
2	$x^2 + x + 1$	3
3	$x^3 + x^2 + 1$	7
4	$x^4 + x^3 + 1$	15
5	$x^5 + x^3 + 1$	31
6	$x^6 + x^5 + 1$	63
7	$x^7 + x^6 + 1$	127
8	$x^8 + x^6 + x^5 + x^4 + 1$	255
9	$x^9 + x^5 + 1$	511
10	$x^{10} + x^7 + 1$	1023
11	$x^{11} + x^9 + 1$	2047
12	$x^{12} + x^{11} + x^{10} + x^4 + 1$	4095
13	$x^{13} + x^{12} + x^{11} + x^8 + 1$	8191
14	$x^{14} + x^{13} + x^{12} + x^2 + 1$	16383
15	$x^{15} + x^{14} + 1$	32767
16	$x^{16} + x^{14} + x^{13} + x^{11} + 1$	65535
17	$x^{17} + x^{14} + 1$	131071
18	$x^{18} + x^{11} + 1$	262143
19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$	524287

More on why modulation is useful

- Discussed channel sharing already
- Now: noise immunity

Noise

Why modulated sensing?

- Johnson noise
  - Broadband thermal noise
- Shot noise
  - Individual electrons…not usually a problem
- "1/f" "flicker" "pink" noise
  - Worse at lower frequencies
  - ➔ do better if we can move to higher frequencies
- 60Hz pickup

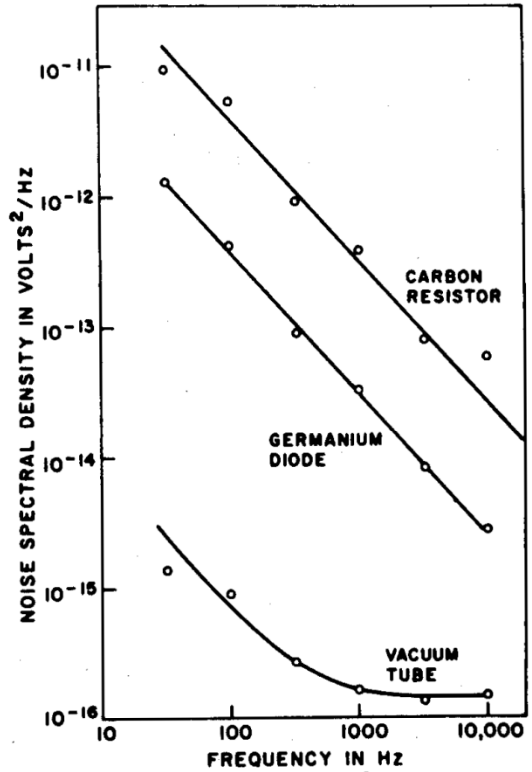


FIGURE 5 Typical electrical noise spectra for some current-carrying devices: 50 KΩ carbon resistor, 2N2000 germanium diode-connected transistor, and 12AX7 vacuum tube. (Reproduced from Brophy).[4]

From W.H. Press, "Flicker noises in astronomy and elsewhere," Comments on astrophysics 7: 103-119. 1978.

Modulation

- What is it?
  - In music, changing key
  - In old time radio, shifting a signal from one frequency to another
  - Ex: voice (10kHz “baseband” sig.) modulated up to 560kHz at radio station
  - Baseband voice signal is recovered when radio receiver demodulates
  - More generally, modulation schemes allow us to use analog channels to communicate either analog or digital information
    - Amplitude Modulation (AM), Frequency Modulation (FM), Frequency hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS), etc
- What is it good for?
  - Sensitive measurements
    - Sensed signal more effectively shares channel with noise ➔ better SNR
  - Channel sharing: multiple users can communicate at once
    - Without modulation, there could be only one radio station in a given area
    - One radio can chose one of many channels to tune in (demodulate)
  - Faster communication
    - Multiple bits share the channel simultaneously ➔ more bits per sec
    - “Modem” == “Modulator-demodulator”

Modulation --- A software perspective

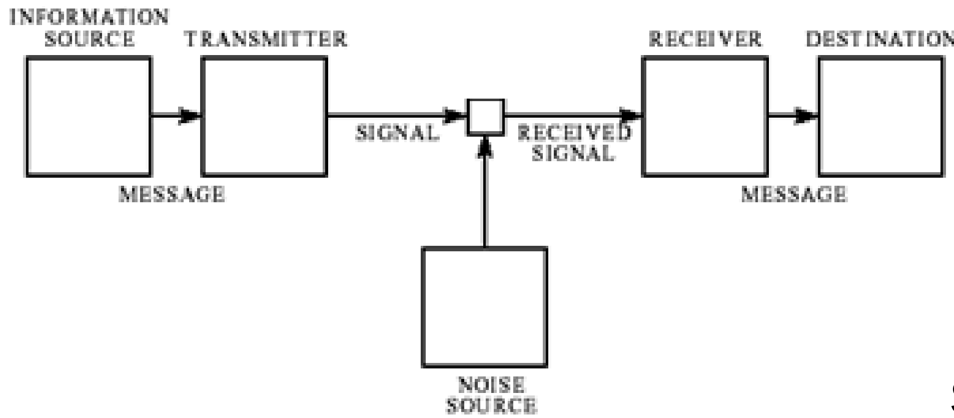


Fig. 1—Schematic diagram of a general communication system.

Shannon

- Q: What determines number of messages we can send through a channel (or extract from a sensor, or from a memory)?
- A: The number of inputs we can reliably distinguish when we make a measurement at the output

Other applications of modulation / demodulation or correlation computations

Other applications of modulation / demodulation or correlation computations

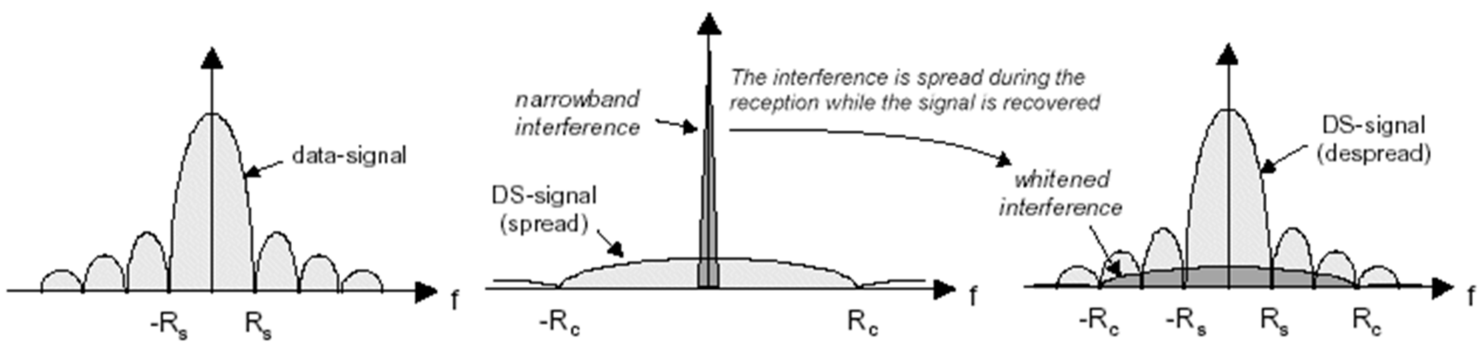
These are extremely useful algorithmic techniques that are not commonly taught or are scattered in computer science

- Amplitude-modulated sensing (what we've been doing)
  - Also known as synchronous detection
- Ranging (GPS, sonar, laser rangefinders)
- Analog RF Communication (AM radio, FM radio)
- Digital Communication (modem==modulator demodulator)
- Data hiding (digital watermarking / steganography)
- Fiber Fingerprinting (biometrics more generally)
- Pattern recognition (template matching, simple gesture rec)

CDMA in comms: Direct Sequence Spread Spectrum (DSSS)

- Other places where DSSS is used
  - 802.11b, GPS
- Terminology
  - Symbols: data
  - Chips: single carrier value
  - Varying number of chips per symbol varies data rate…when SNR is lower, increase number of chips per symbol to improve robustness and decrease data rate
  - Interference: one channel impacting another
  - Noise (from outside)

Visualizing DSSS



https://www.okob.net/texts/mydocuments/80211physlayer/images/dsss_interf.gif

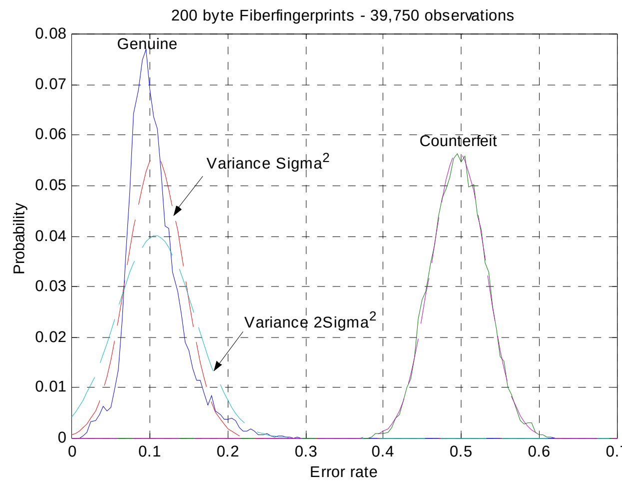
Practical DSSS radios

- DSSS radio communication systems in practice use the pseudo-random code to modulate a sinusoidal carrier (say 2.4GHz)
- This spreads the energy somewhat around the original carrier, but doesn't distribute it uniformly over all bands, 0-2.4GHz
- Amount of spreading is determined by chip time (smallest time interval)

Data hiding

“Modulation and Information Hiding in Images,” Joshua R. Smith and Barrett O. Comiskey. Presented at the Workshop on Information Hiding, Isaac Newton Institute, University of Cambridge, UK, May 1996; Springer-Verlag Lecture Notes in Computer Science Vol. 1174, pp 207-226.

FiberFingerprint



FiberFingerprint Identification
Proceedings of the Third Workshop on Automatic Identification, Tarrytown, NY, March 2002
E. Metois, P. Yarin, N. Salzman, J.R. Smith

Key in this application: remove DC component before correlating

Gesture recognition by cross-correlation of sensor data with a template

$$C = \sum_{i=1}^{n} (A_{x_i} T_{x_i} + A_{y_i} T_{y_i} + A_{z_i} T_{z_i})$$

"RFIDs and Secret Handshakes: Defending Against Ghost-and-Leech Attacks and Unauthorized Reads with Context-Aware Communications,"
A. Czeskis, K. Koscher, J.R. Smith, and T. Kohno
15th ACM Conference on Computer and Communications Security (CCS), Alexandria, VA.
October 27-31, 2008

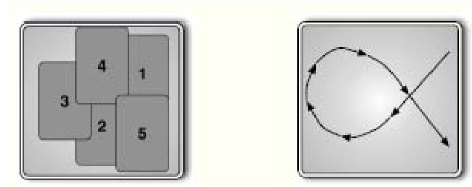


Figure 2: Example secret handshake/activation scheme. Both images show the alpha (α) motion performed with the card in front of the reader. In the left image, numbers indicate sequence of card positions across reader with time. In the right image, arrows show how the card moves across the reader with time.

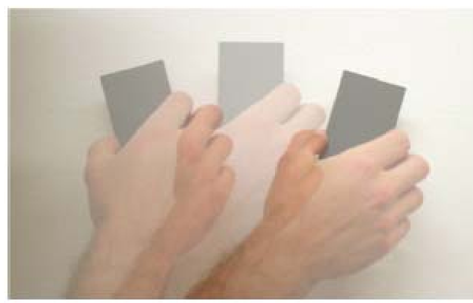


Figure 3: Example secret handshake/activation scheme. In this image, we demonstrate the 1.5-wave gesture.

Figure 4: Our prototype WISP RFID system enclosed in a custom plastic box. Wires lead out the back for debugging and tethered experiments. Box cover shown at bottom left. Next to it is a standard HID proximity card.

age scenario — the user is in possession of an RFID access card, which he must present to the RFID reader to gain access to a protected resource. In accordance with our design discussion in Section 3, we performed context detection on the RFID access card.

Limitations

- TX and RCV need common time-scale (or length scale)
  - Will not recognize a gesture being performed at a different speed than the template
- Except in sensing (synchronous detection) applications, need to synchronize TX and RX…this is a search that can take time

End of section