Communications Lab
ECEN 4652/5002, Lecture 1

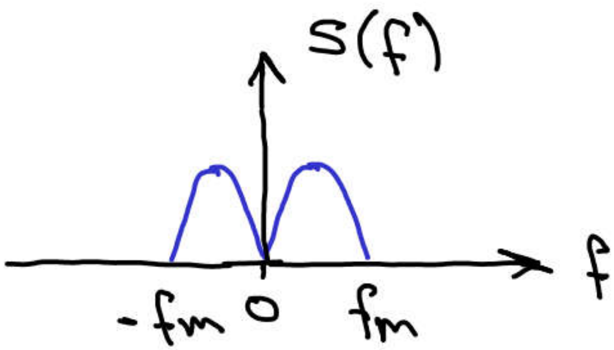
Peter Mathys

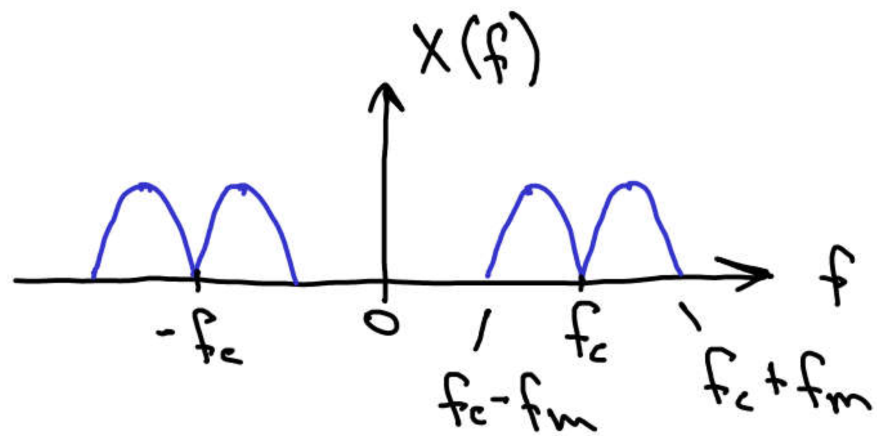
Goals of Lab 1

- Use Python as a high-level tool for signal processing.
- Transmit and receive ASCII text strings using “flat-top PAM” (pulse amplitude modulation).
- Use PCM (pulse code modulation) to transmit analog signals using flat-top PAM.
- Distinguish between DT (discrete time), CT/”pseudo CT” (continuous time) signals.

Baseband vs Bandpass Signals

- Baseband signal: Filter of smallest bandwidth (BW) that passes signal is a LPF.
- Bandpass signal: Filter of smallest BW that passes signal is BPF.





Fourier Transform Properties

- Real x(t), frequency shift, time shift

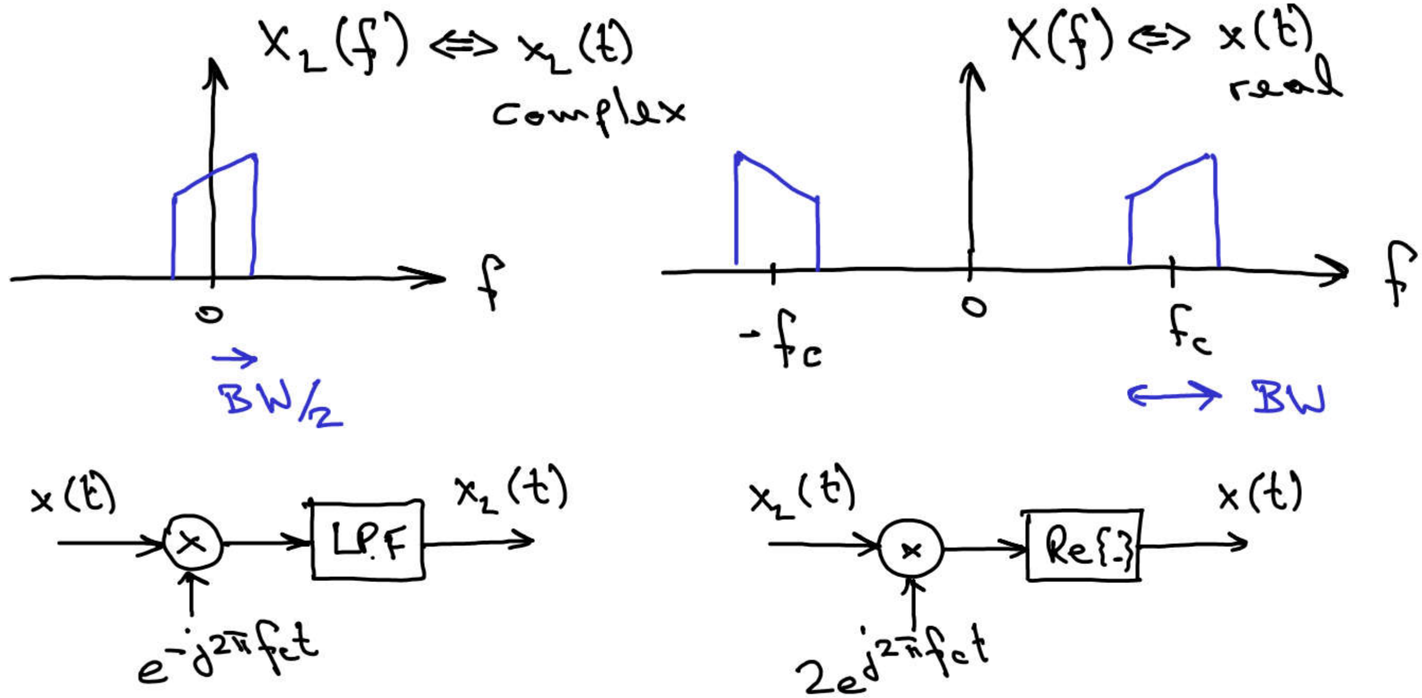
$$x(t)\ \text{real} \quad \Leftrightarrow \quad \begin{array}{l} |X(f)| = |X(-f)| \\ \angle X(f) = -\angle X(-f) \end{array}$$

$$x(t)\, e^{-j2\pi f_c t} \quad \Leftrightarrow \quad X(f + f_c)$$

$$x(t-\tau) \quad \Leftrightarrow \quad X(f)\, e^{-j2\pi f \tau}$$

Complex Baseband vs Real Bandpass

- Real-valued bandpass signals can be uniquely represented as complex baseband signals.



$$x_L(t) = x_I(t) + j\, x_Q(t)$$

I: in-phase, Q: quadrature component

7-Bit ASCII (American Standard Code for Information Interchange)								
	000...	001...	010...	011...	100...	101...	110...	111...
..0000	*NUL*	*DLE*	*SP*	0	@	P	`	p
..0001	*SOH*	*DC1*	!	1	A	Q	a	q
..0010	*STX*	*DC2*	"	2	B	R	b	r
..0011	*ETX*	*DC3*	#	3	C	S	c	s
..0100	*EOT*	*DC4*	$	4	D	T	d	t
..0101	*ENQ*	*NAK*	%	5	E	U	e	u
..0110	*ACK*	*SYN*	&	6	F	V	f	v
..0111	*BEL*	*ETB*	'	7	G	W	g	w
..1000	*BS*	*CAN*	(	8	H	X	h	x
..1001	*HT*	*EM*	)	9	I	Y	i	y
..1010	*LF*	*SUB*	*	:	J	Z	j	z
..1011	*VT*	*ESC*	+	;	K	[	k	{
..1100	*FF*	*FS*	,	<	L	\	l	|
..1101	*CR*	*GS*	-	=	M	]	m	}
..1110	*SO*	*RS*	.	>	N	^	n	~
..1111	*SI*	*US*	/	?	O	_	o	*DEL*

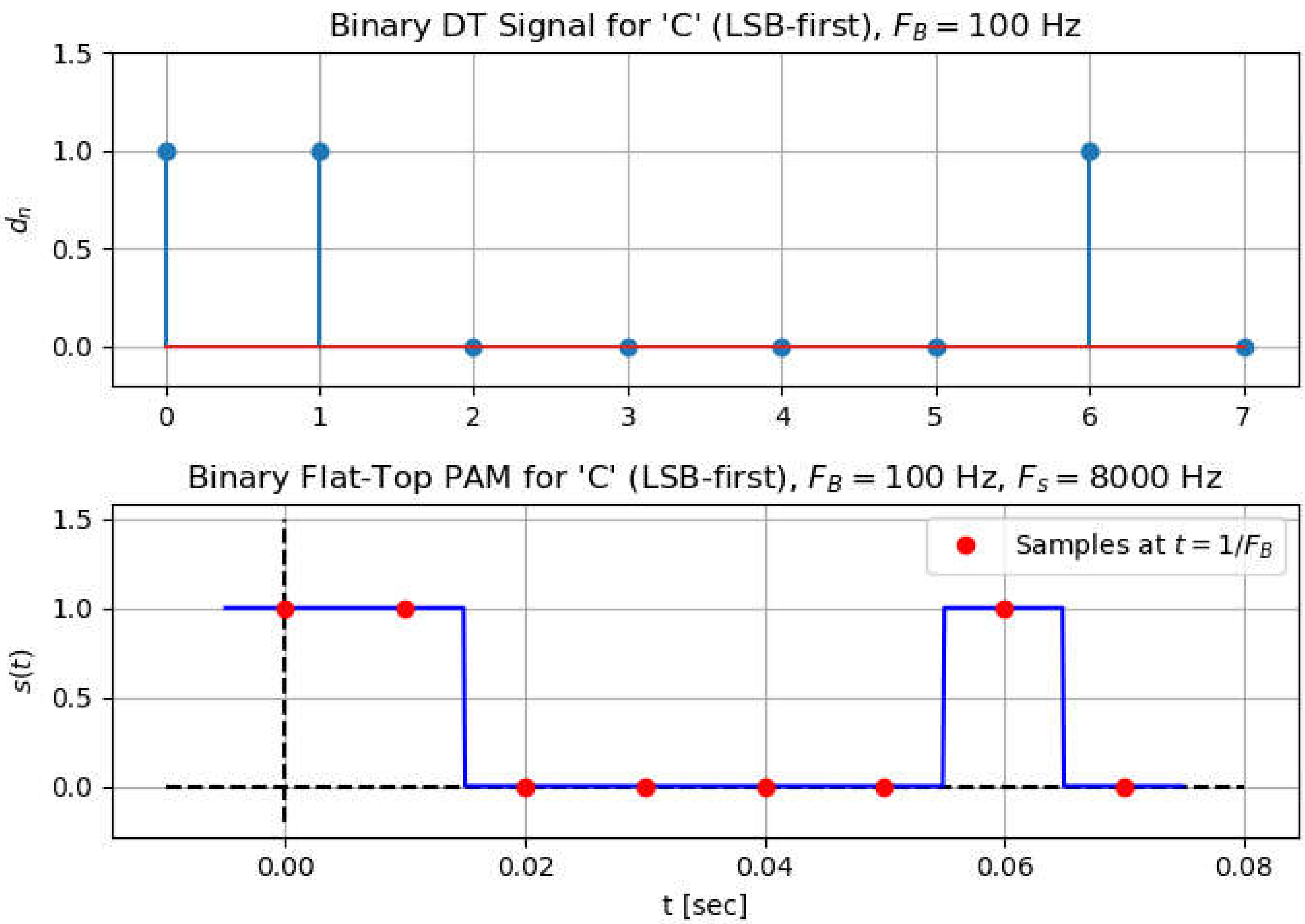
Parallel to Serial Conversion

"Test" in Extended (8-bit) ASCII	
Character	**Extended ASCII Code**
T	01010100
e	01100101
s	01110011
t	01110100

MSB-first Bit Sequence for "Test" (Extended 8-bit ASCII)
$d_n = 01010100\ 01100101\ 01110011\ 01110100$ $\rightarrow$ Index n increases from left to right $\rightarrow$
d_0=0, d_1=1, d_2=0, d_3=1, d_4=0, d_5=1, d_6=0, d_7=0, d_8=0, d_9=1, d_{10}=1, ...

LSB-first Bit Sequence for "Test" (Extended 8-bit ASCII)
$d_n = 00101010\ 10100110\ 11001110\ 00101110$ $\rightarrow$ Index n increases from left to right $\rightarrow$
d_0=0, d_1=0, d_2=1, d_3=0, d_4=1, d_5=0, d_6=1, d_7=0, d_8=1, d_9=0, d_{10}=1, ...

Binary Flat-Top PAM



C = 01000011

Stem Plot for DT Signal

Flat-Top PAM Is CT Signal

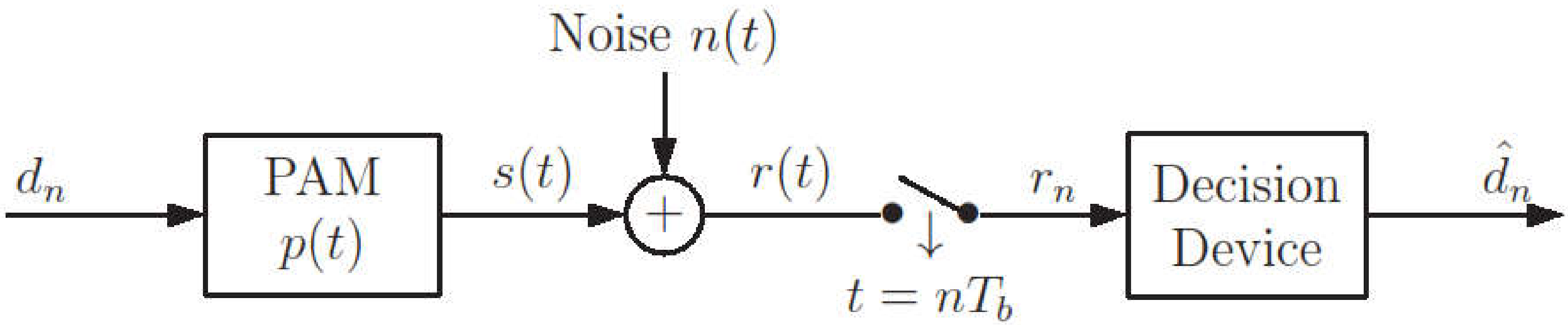
CT versus DT Signals in Python

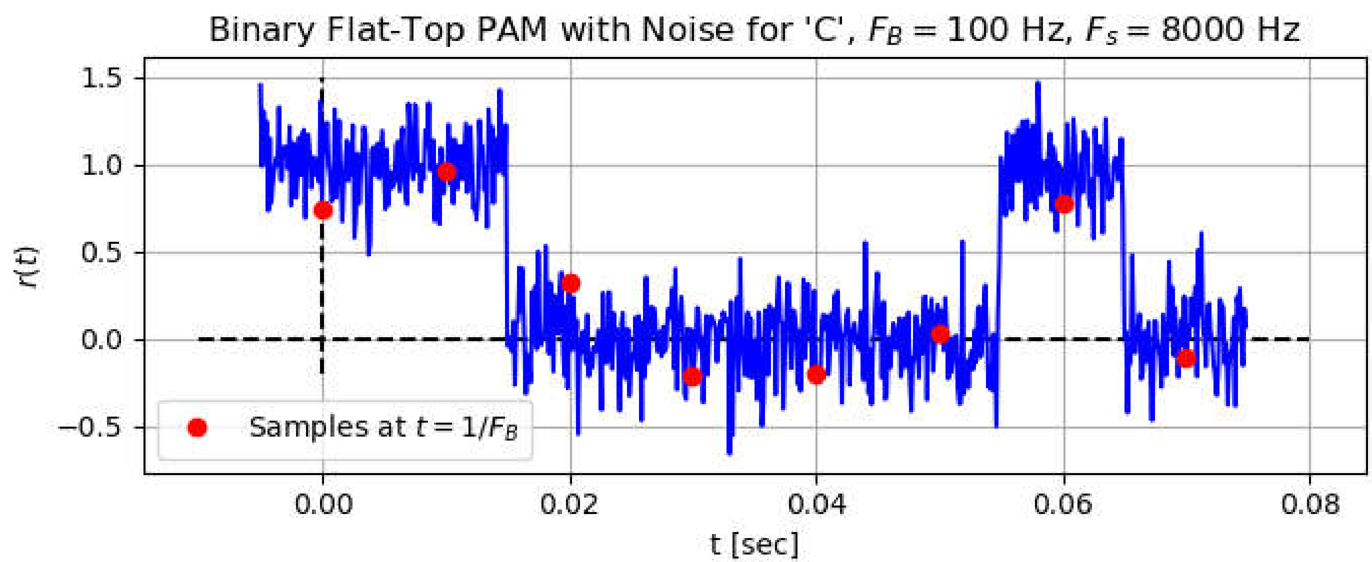
- The serial representation of an ASCII character is a DT vector $(d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7)$
- The Flat-Top PAM is conversion from DT to CT through the CT pulse p(t)

$$s(t) = \sum_n d_n \, p(t - nT_B)$$

- In Python we can only use “pseudo CT” signals, i.e., DT signals with a much higher sampling rate than the DT symbol rate (baud rate F_B=1/T_B)

Reception of Noisy Signal



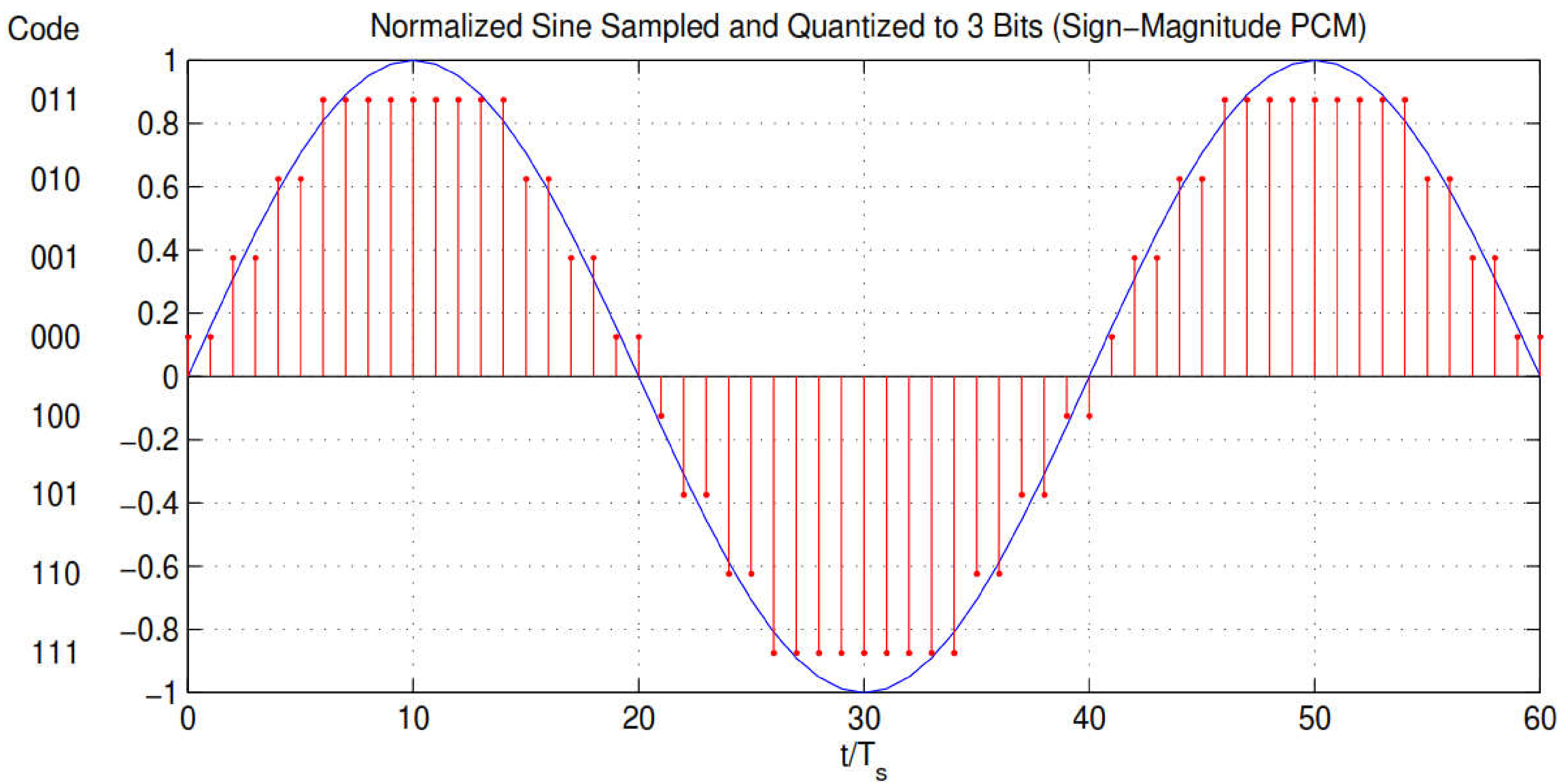


Using single sample per symbol not optimal

Combat Noise for Analog Signals

- PCM: Pulse Code Modulation
- Sample analog signal at rate F_s and quantize amplitude to n bits ($L=2^n$ levels)
- Then use parallel to serial conversion for each binary n-bit word
- The resulting DT sequence (symbol rate $F_B=n*F_s$) is then converted to a CT flat-top PAM signal and transmitted

Example: 3-bit PCM for Sinewave



Example: 3-bit PCM for Sinewave