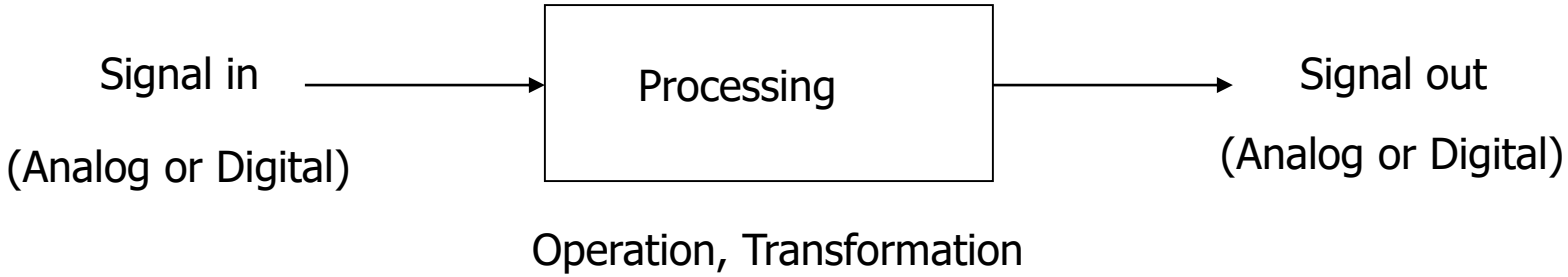
Digital Signal Processing

Arak University of Technology

By: Dr. Moein Ahmadi

What is Signal Processing?

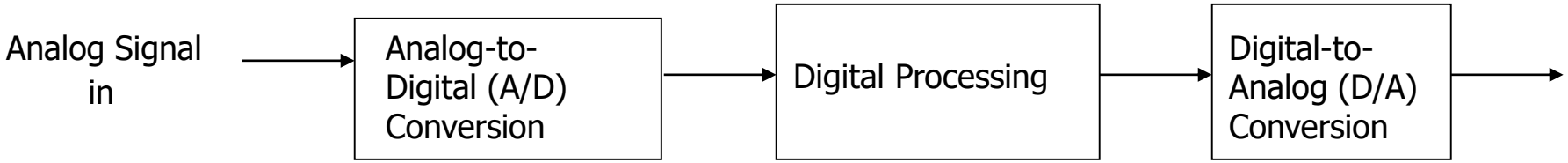


- Example of Signals:
  - Analog: Speech, Music, Photos, Video, Radar, Sonar, …
  - Discrete-domain/Digital:
    - digitized speech, digitized music, digitized images, digitized video, digitized radar and sonar signals,…
    - stock market data, daily max temperature data, …

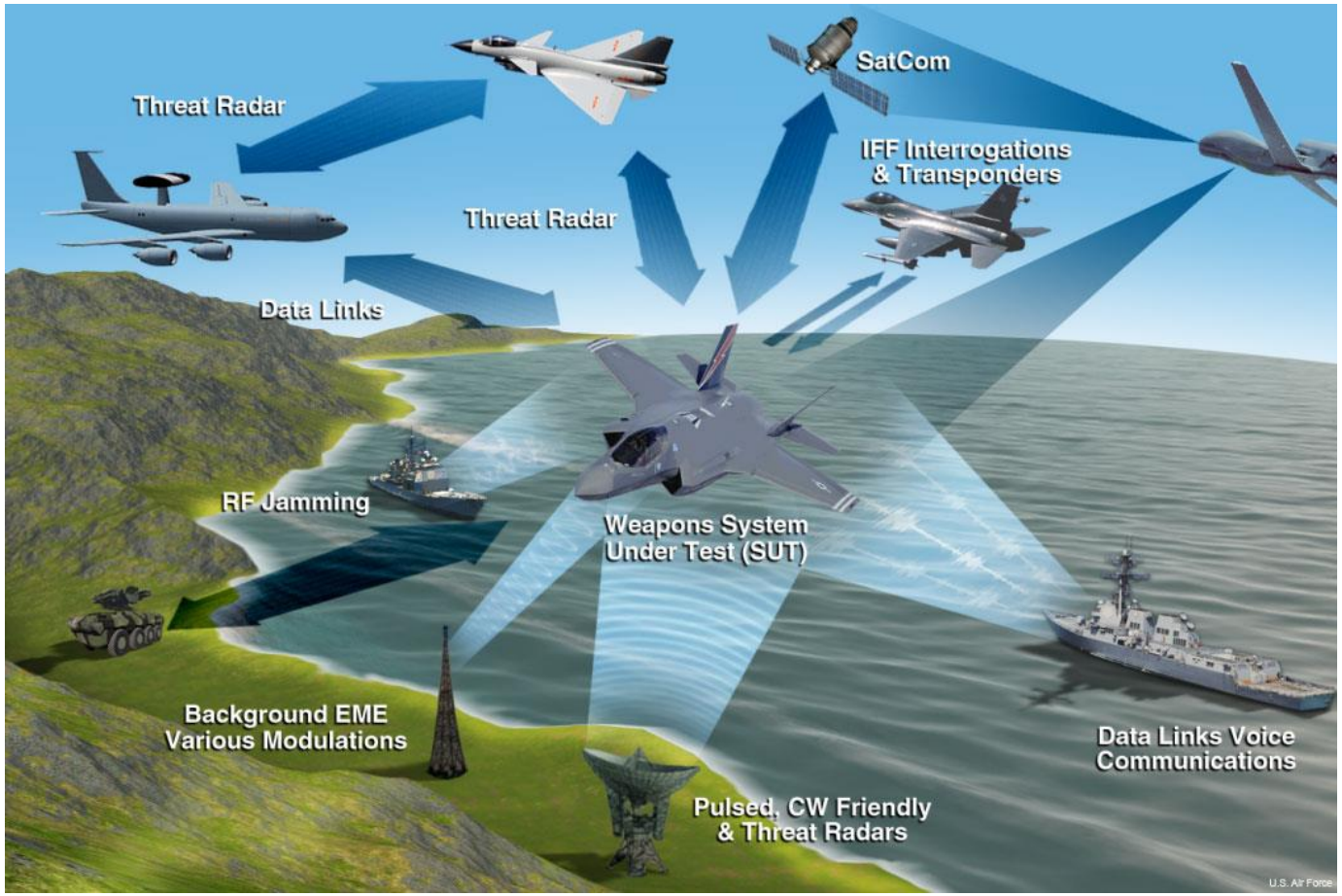
Digital Signal Processing and ADC

Operation, Transformation performed on digital signals (using a computer or other special-purpose digital hardware)

- But what about analog signals?



General System- EW Environment



رادار
شنود
جمینگ
مخابرات
سنسور های مختلف

رادار: جستجو – رهگیر – تصویر برداری

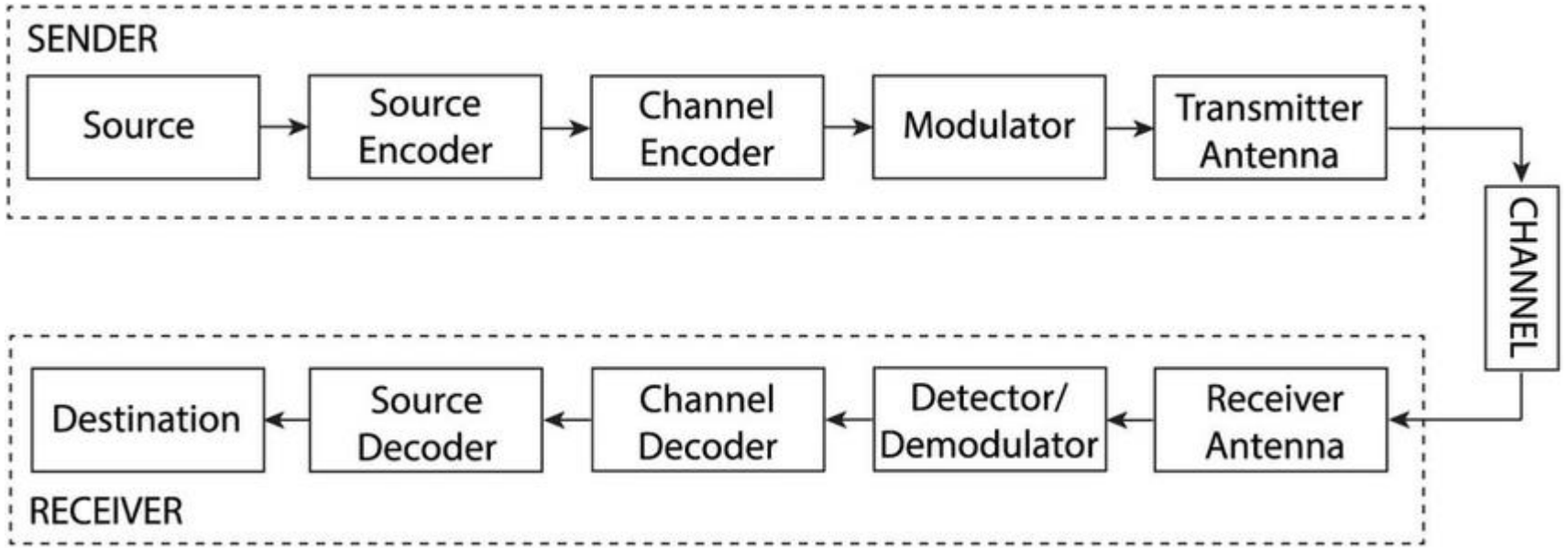
رادار جستجو:

آشکارسازی و تعیین فاصله و سرعت اهداف

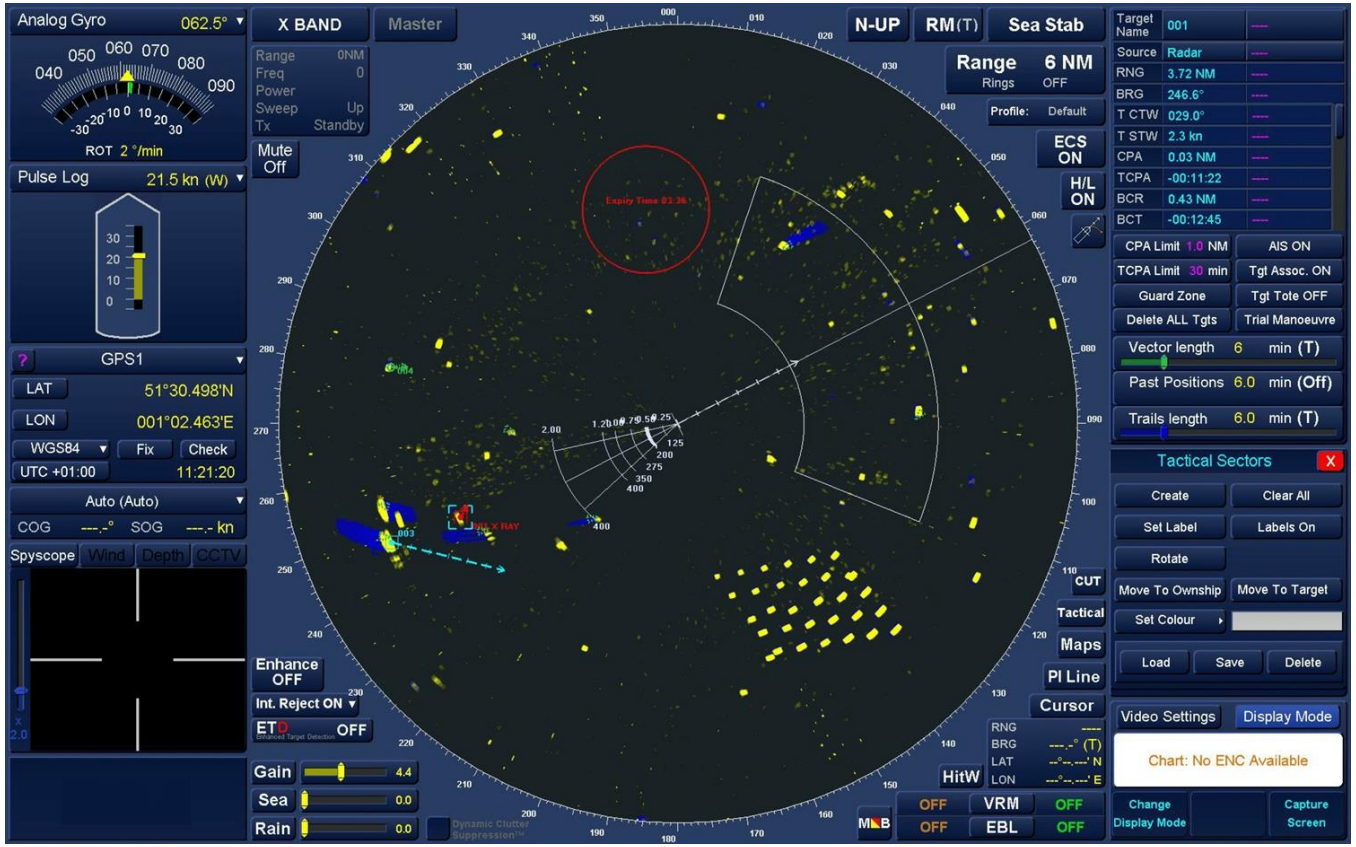
فیلتر فشرده سازی
پردازش داپلر
آشکارسازی
رهگیری حین اسکن

نمونه هایی از سیستم های مورد بحث
در پردازش سیگنال

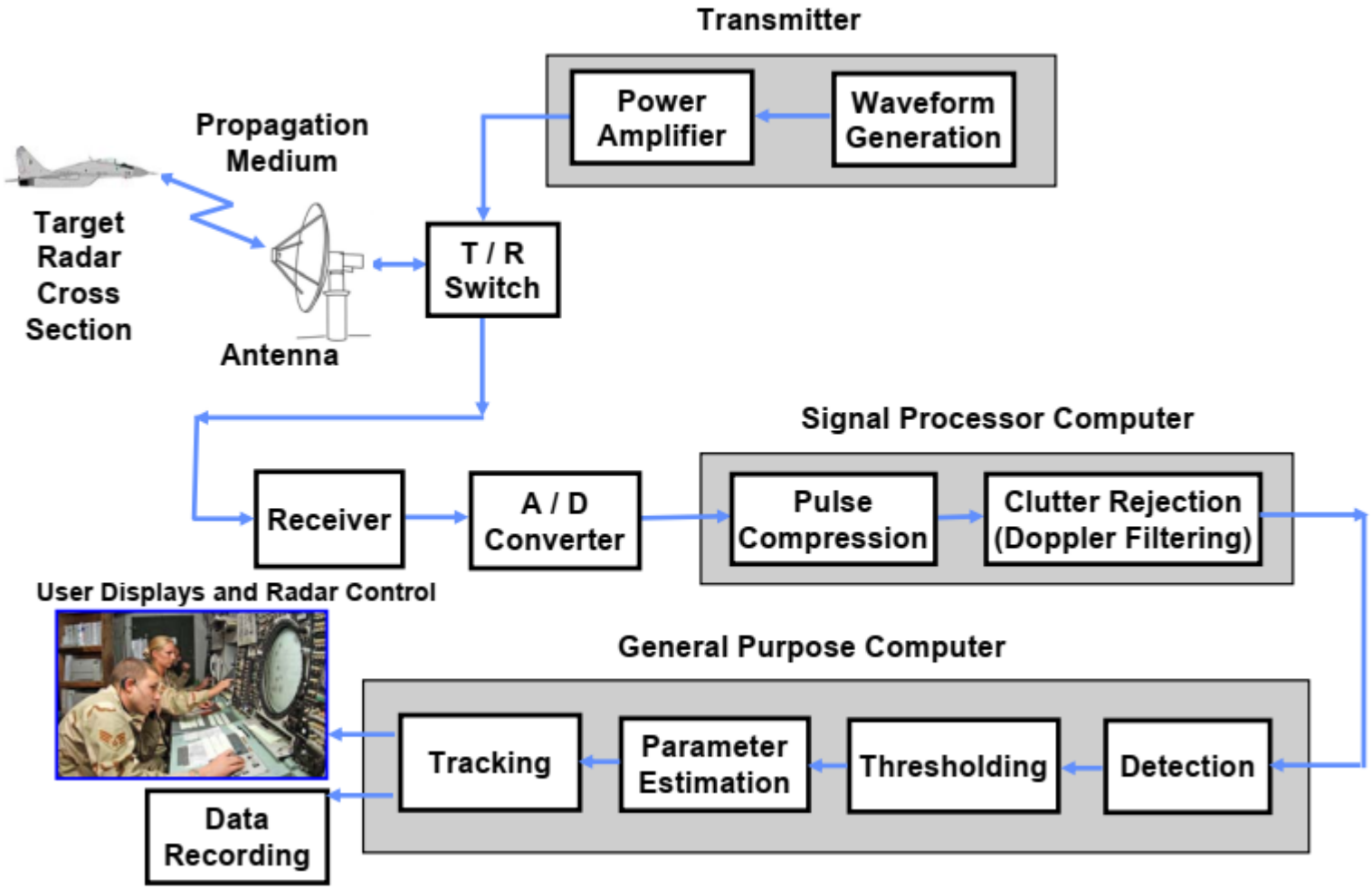
Communications System Block Diagram



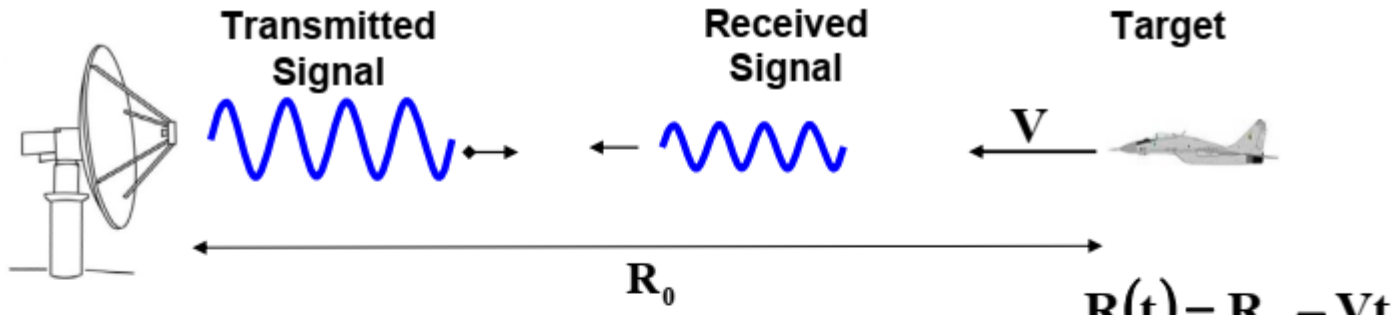
General System- Phased Array Radar



Example- Radar System



Example- Radar Basics



Transmitted Signal: $s_T(t) = A(t)\exp(j2\pi f_0 t)$

Received Signal: $s_R(t) = \alpha A(t-\tau)\exp[j2\pi(f_0 + f_D)t]$

Amplitude

Depends on RCS, radar parameters, range, etc.

Angle

Azimuth and Elevation

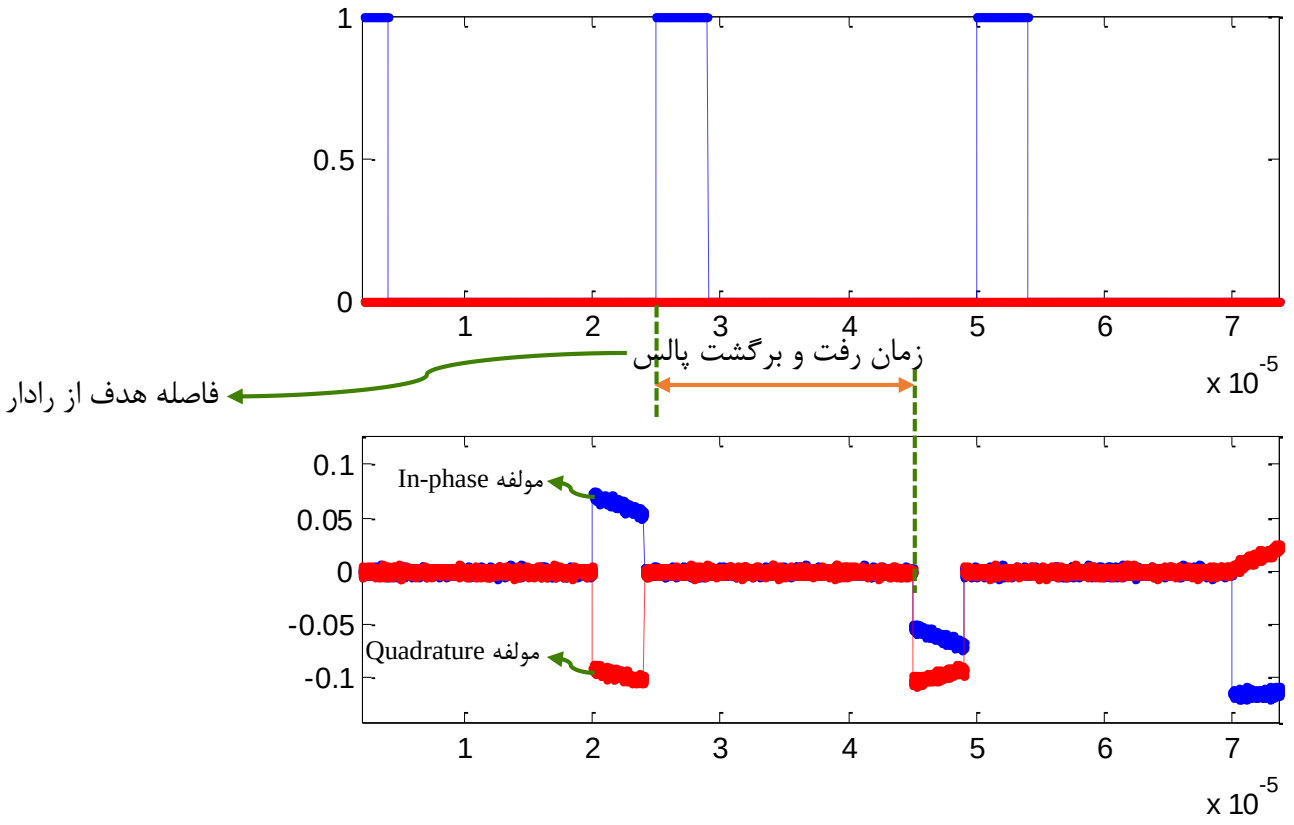
Time Delay

$$\tau = \frac{2R_0}{c}$$

Doppler Frequency

$$f_D = \frac{2Vf_0}{c} = \frac{2V}{\lambda}$$

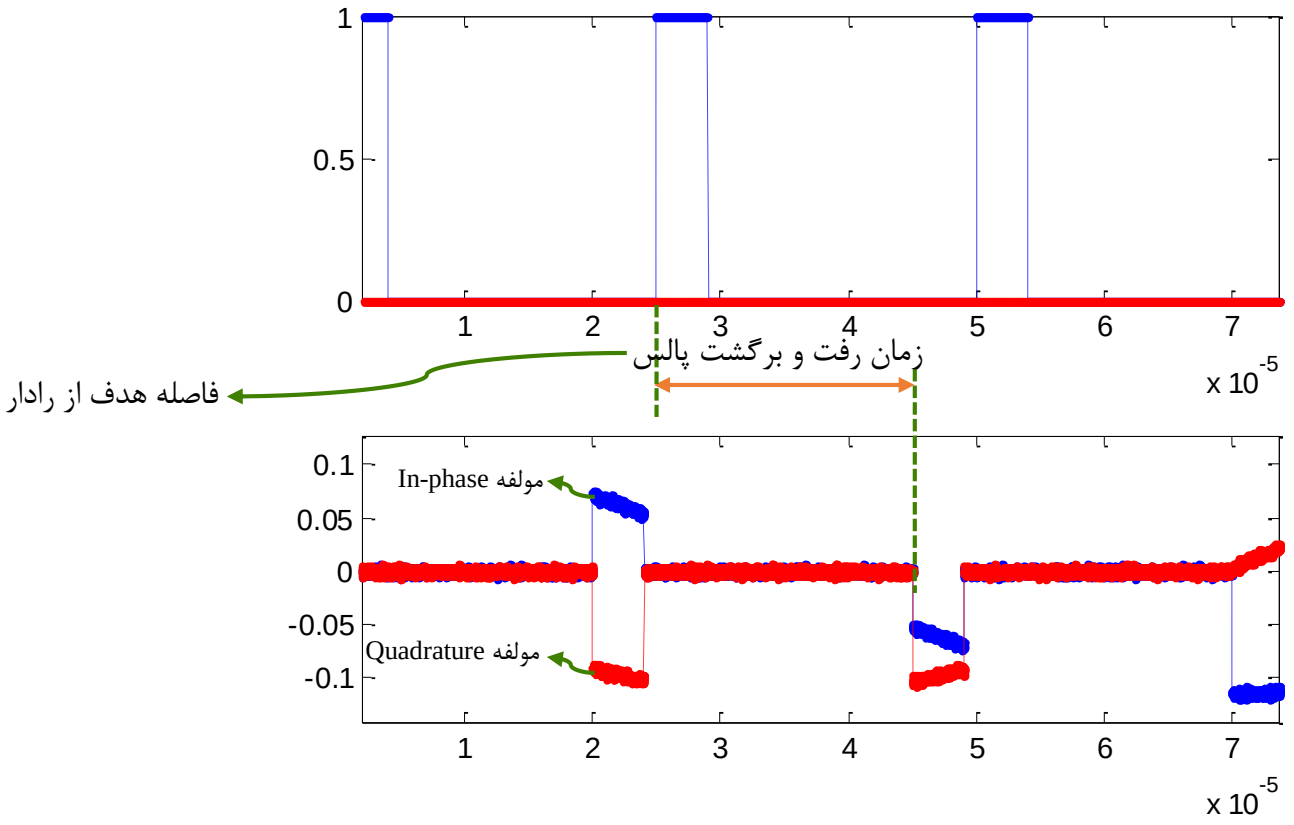
Radar Basics



سیگنال ارسالی
از فرستنده رادار

سیگنال دریافتی از
هدف در گیرنده رادار

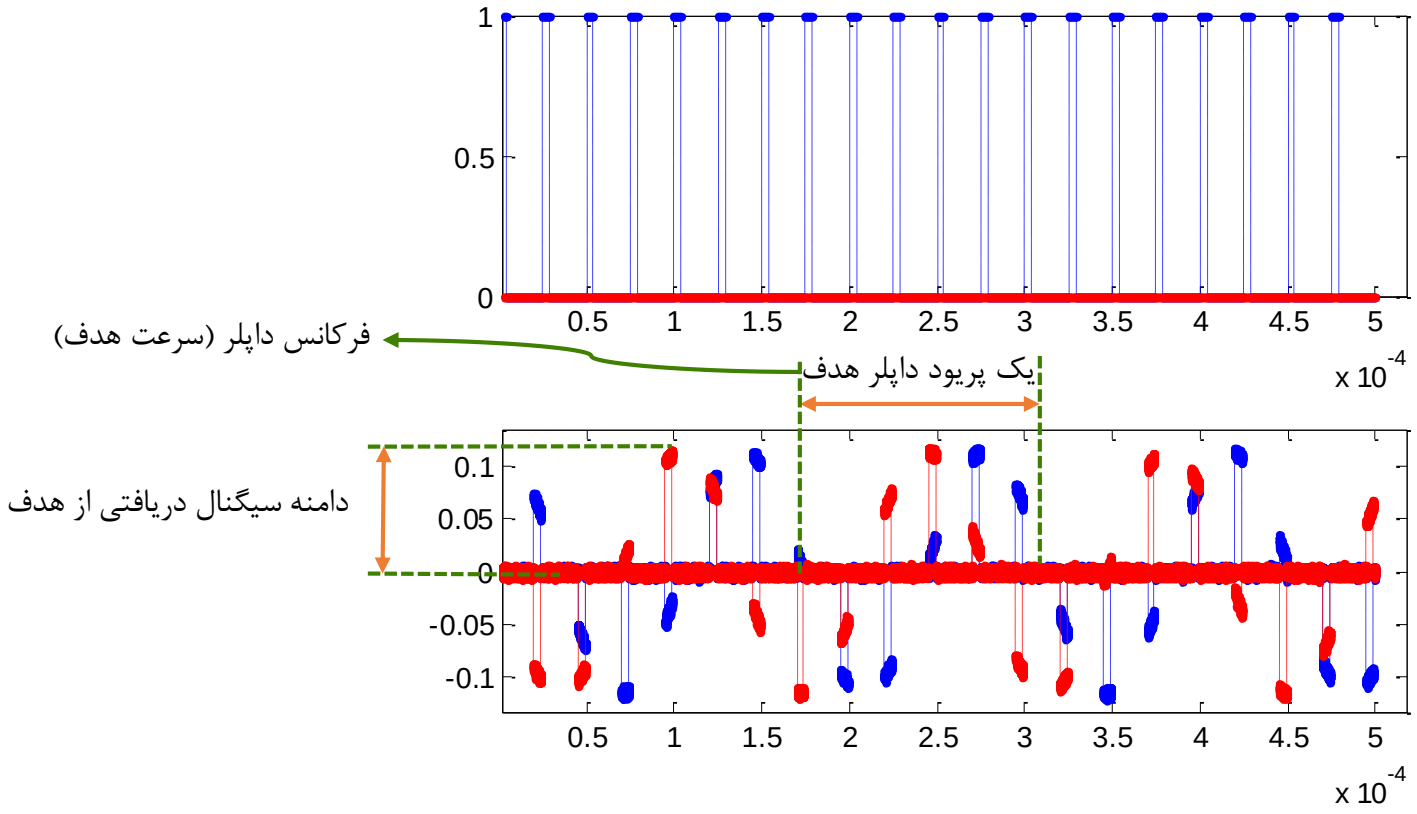
Radar Basics



سیگنال ارسالی
از فرستنده رادار

سیگنال دریافتی از
هدف در گیرنده رادار

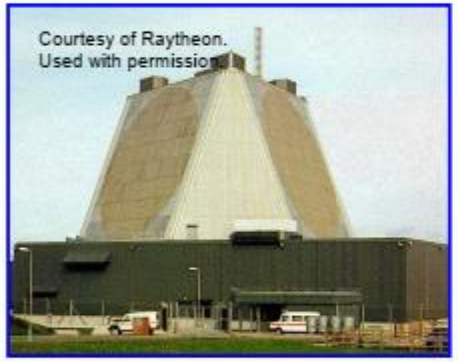
Radar Basics



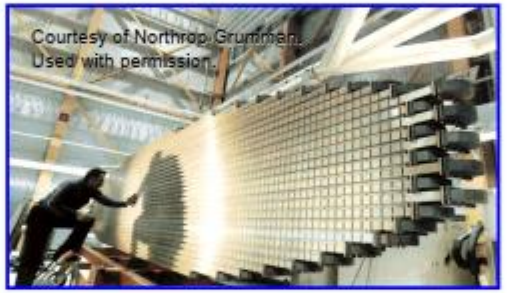
سیگنال ارسالی
از فرستنده رادار

سیگنال دریافتی از
هدف در گیرنده رادار

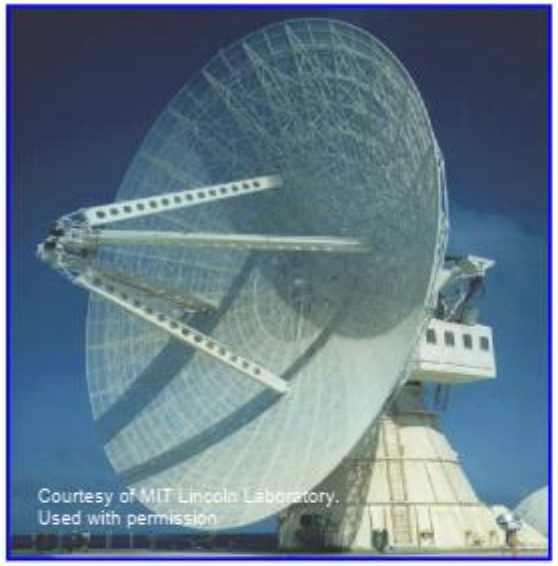
Surveillance and Fire Control Radars

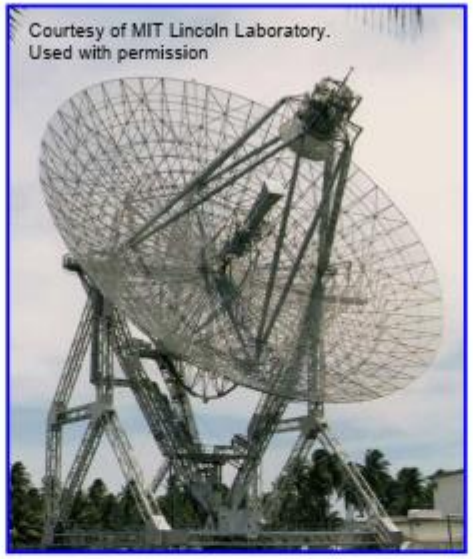


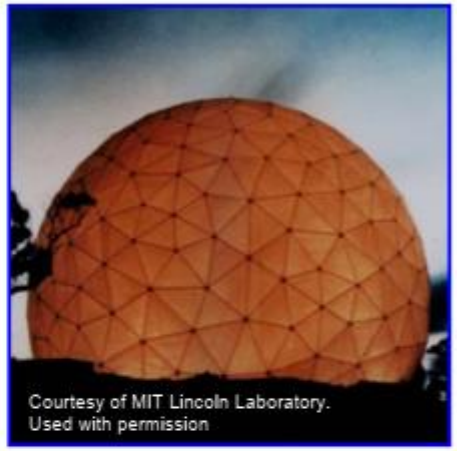
Airborne Radars



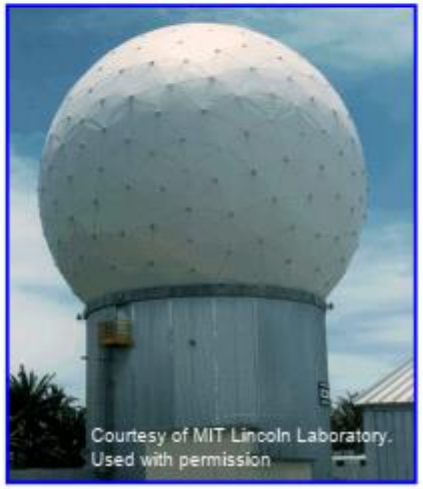
Instrumentation Radars



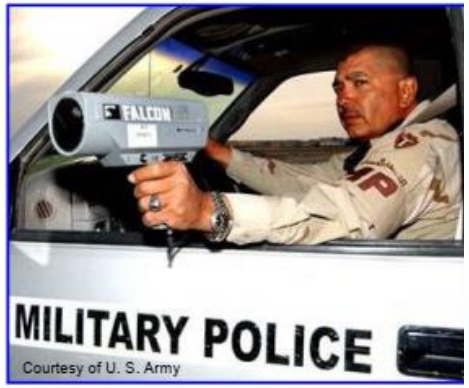




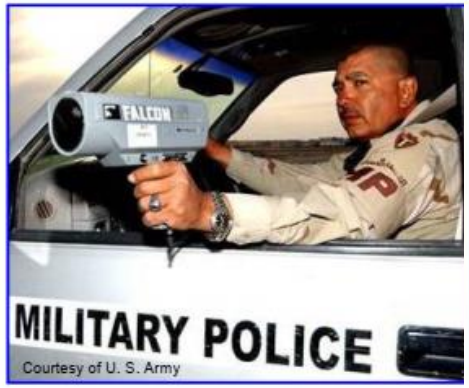
Courtesy of Lockheed Martin



Civil Radars



Civil Radars



Radars in Iran-Hawk

Radars in Iran-Nebo

Signal Processing Topics

Audio and Acoustic Signal Processing	Quantum Signal Processing
Biomedical Signal and Image Processing	Remote Sensing and Signal Processing
Compressive Sensing, Sampling, and Dictionary Learning	Sensor Array & Multichannel Signal Processing
Design and Implementation of Signal Processing	Signal Processing for Big Data
Financial Signal Processing	Spoken language processing
Graph Theory and Signal Processing	Signal Processing for Communication and Networking
Image, Video and Multidimensional	Signal Processing for Cyber Security
Industrial Signal Processing	Signal Processing for Education
Information Forensics and Security	Signal Processing for Smart Systems
Internet of Things & RFID	Signal Processing Implementation
Machine Learning for Signal Processing	Signal Processing Theory and Methods
Multimedia Signal Processing	Speech Processing

Signal Processing Topics

- Audio and acoustic signal processing
- Speech and language processing
- Image and video processing
- Multimedia signal processing
- Signal processing theory and methods
- Sensor array and multichannel signal processing
- Signal processing for communications

- Signal processing for education
- Bioinformatics and genomics
- Signal processing for big data
- Signal processing for the internet of things
- Design/implementation of signal processing systems

- Radar and sonar signal processing
- Signal processing over graphs and networks
- Nonlinear signal processing
- Statistical signal processing
- Compressed sensing and sparse modelling
- Optimization methods

- Machine learning
- Bio-medical image and signal processing
- Signal processing for computer vision and robotics
- Computational imaging / spectral imaging
- Information forensics and security
- Signal processing for power systems

EUSIPCO 2019
27th European Signal Processing Conference
A Coruña Spain, September 2-6, 2019

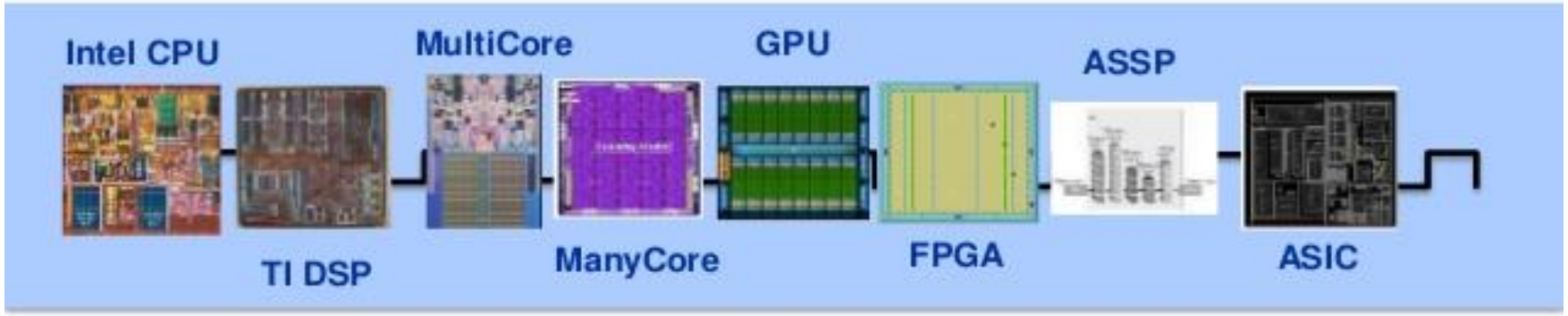
Signal Processing Topics

- Adaptive beamforming
- Array processing for biomedical applications
- Array processing for communications
- Big data
- Blind source separation and channel identification
- Computational and optimization techniques
- Compressive sensing and sparsity-based signal processing
- Detection and estimation
- Direction-of-arrival estimation
- Distributed and adaptive signal processing
- Intelligent systems and knowledge-based signal processing
- Microphone and loudspeaker array applications
- MIMO radar
- Multi-antenna systems: multiuser MIMO, massive MIMO and space-time coding
- Multi-channel imaging and hyperspectral processing
- Multi-sensor processing for smart grid and energy
- Non-Gaussian, nonlinear, and non-stationary models
- Optimization techniques
- Performance evaluations with experimental data
- Radar and sonar array processing
- Sensor networks
- Source Localization, classification and tracking
- Synthetic aperture techniques
- Space-time adaptive processing
- Statistical modelling for sensor arrays
- Tensor signal processing
- Waveform diverse sensors and systems

Related Courses

- Advanced DSP
- Statistical Signal Processing
- Adaptive Filters
- Blind Source Separation and Sparsity-aware Signal Processing
- Detection and Estimation
- Spectrum Estimation
- Radar Systems, Phased-Array and MIMO Radar
- Array Processing, Direction Finding, Interference Nulling

DSP Implementation Hardware



Flexibility, Programming Abstraction

Performance, Area and Power Efficiency

CPU:
- Market-agnostic
- Accessible to many programmers (C++)
- Flexible, portable

FPGA:
- Somewhat Restricted Market
- Harder to Program (Verilog)
- More efficient than SW
- More expensive than ASIC

ASIC
- Market-specific
- Fewer programmers
- Rigid, less programmable
- Hard to build (physical)

ALTERA.

Programming Languages for Signal Processing Course (Algorithm Design)

C++

python™

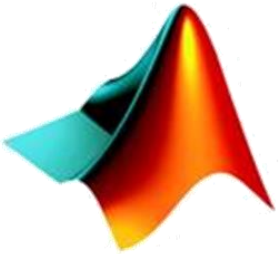
MATLAB

Qt

jupyter

spyder

ANACONDA



MATLAB

Anaconda Pip conda

Python 2.7 and 3.5.

```
python --version
```

Basic data types:

Numbers, Booleans, Strings

Containers:

Lists [,,]

Dictionaries {“:”,”:”}

Sets {'', ‘’}

Tuples (‘’,’’)

Slicing, Loops

Functions

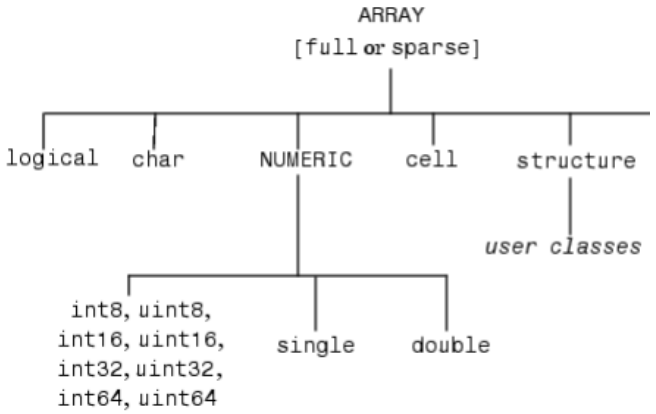
Classes

Libraries Numpy SciPy

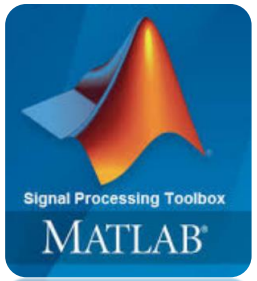
Matplotlib

https://www.mathworks.com

Basic data types:



Toolbox



https://download.qt.io/archive/qt/5.12/5.12.1/

Basic data types:

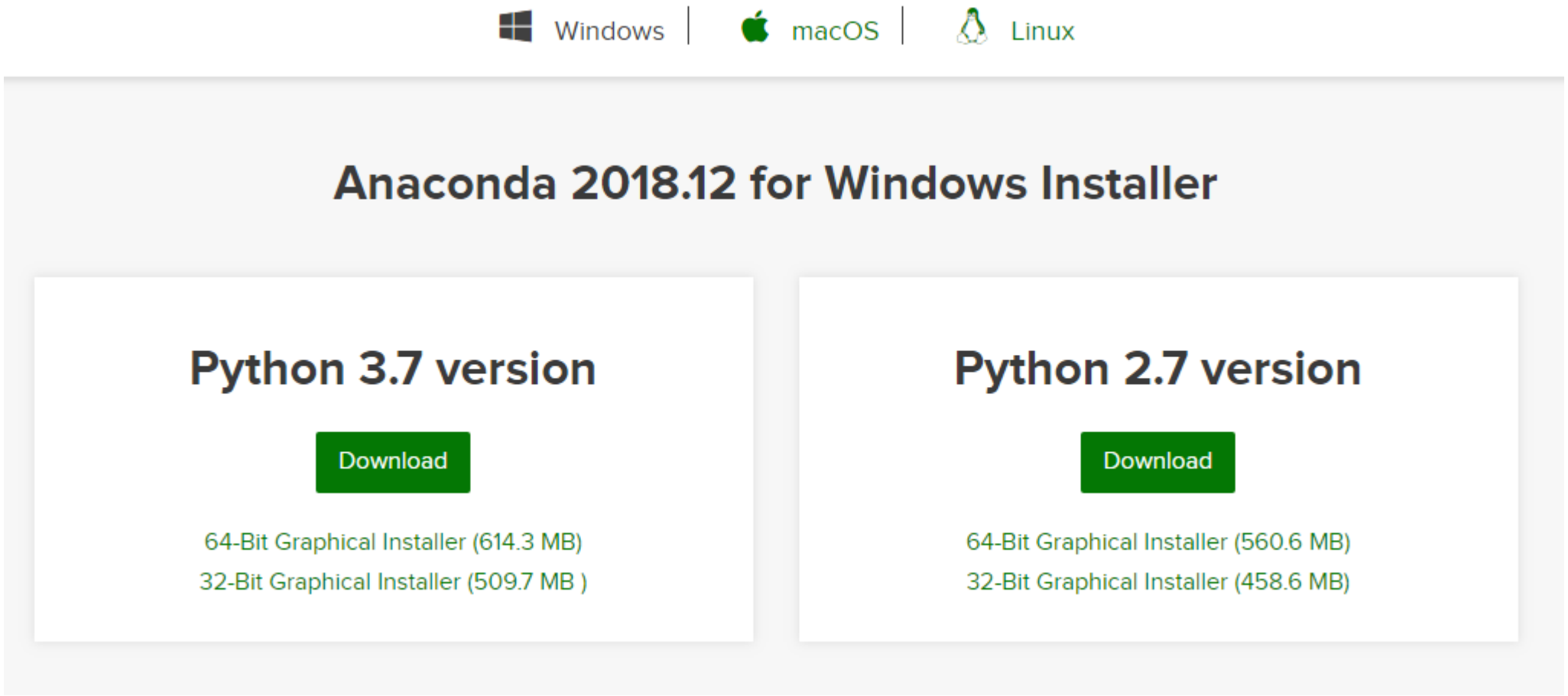
Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,648 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

Containers: QVector , …

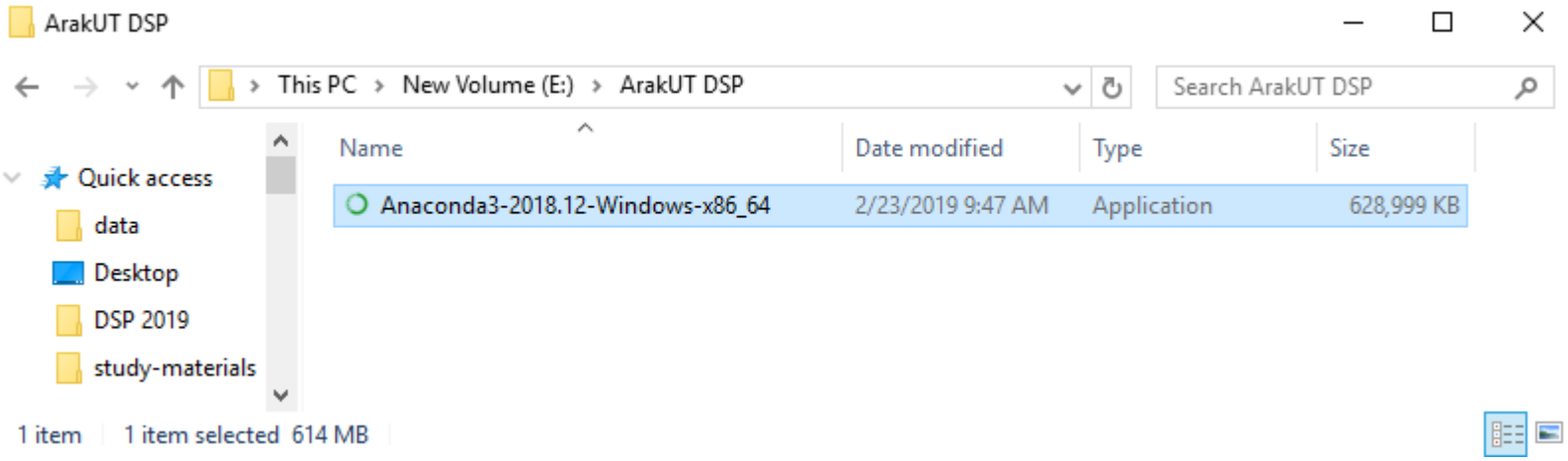
Classes

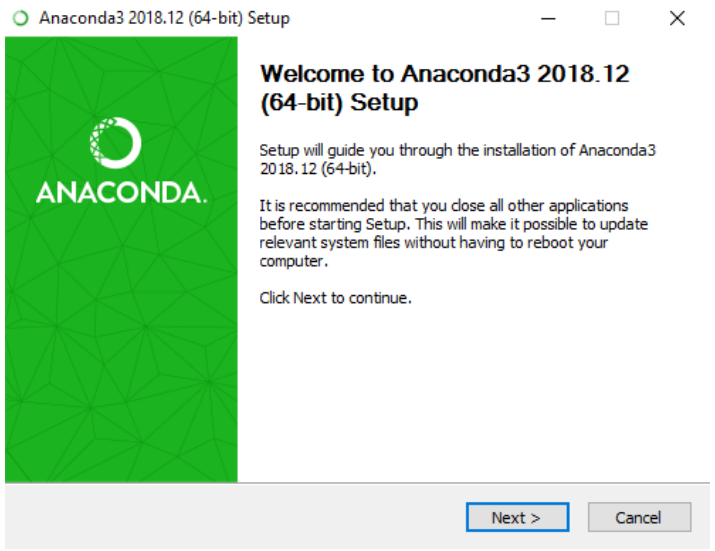
Libraries boost

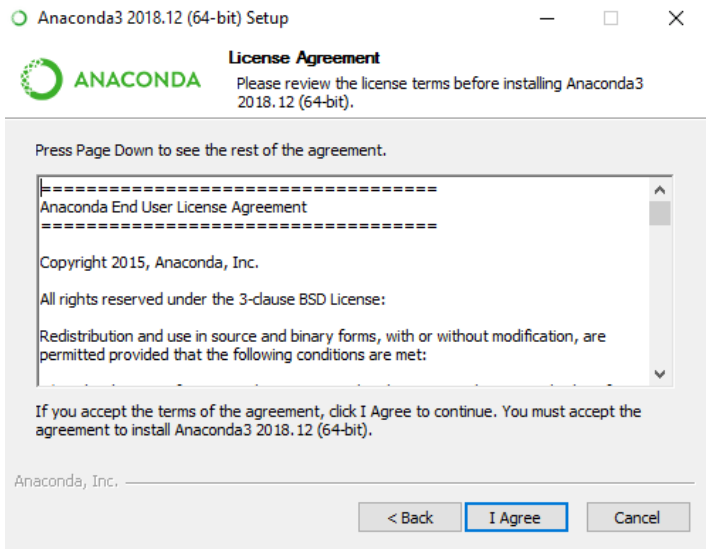
Getting Started With Python Using Anaconda

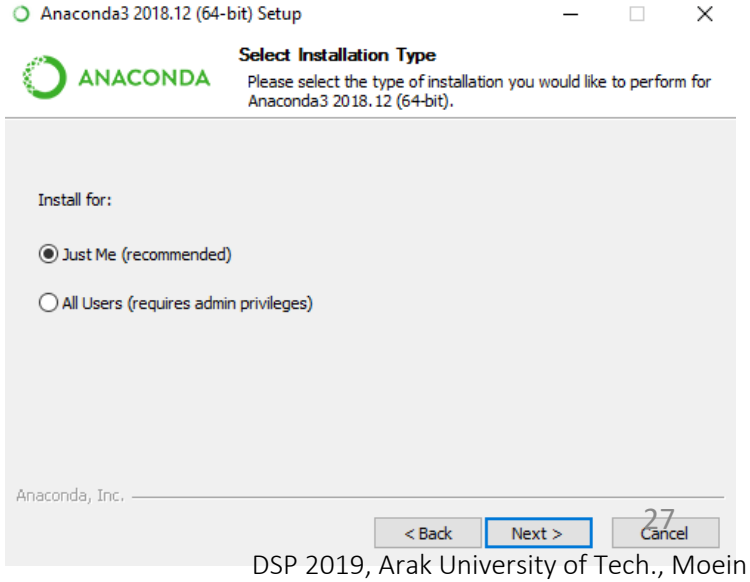


Getting Started With Python Using Anaconda

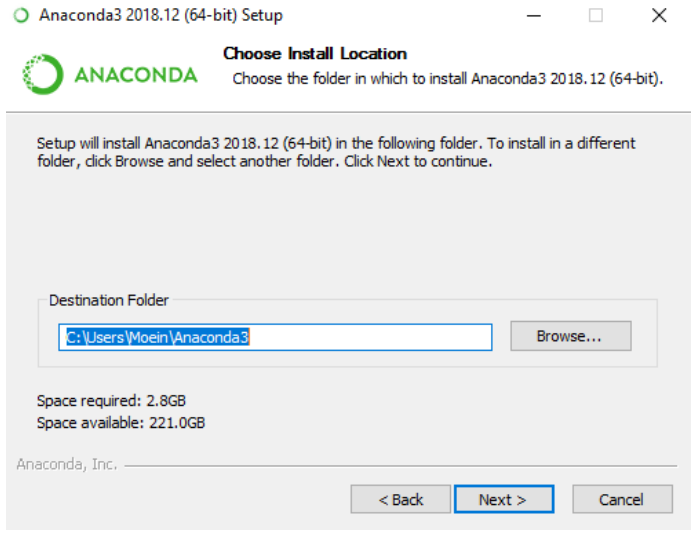


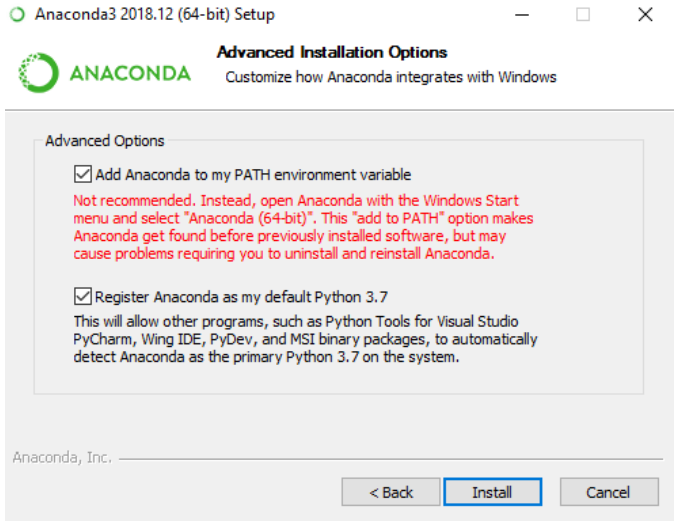


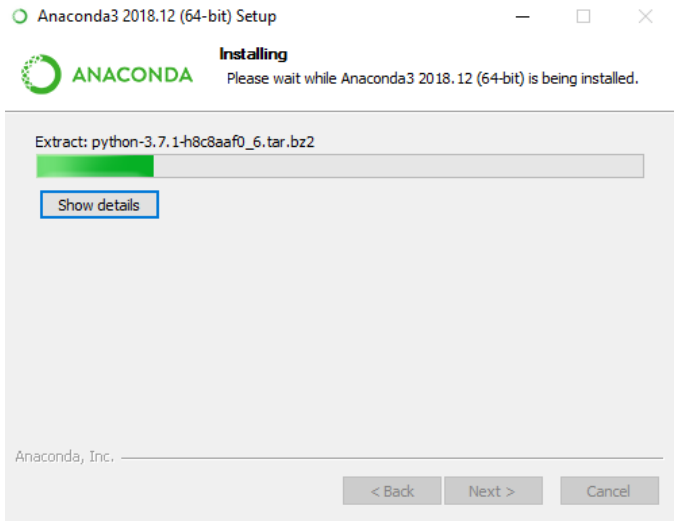


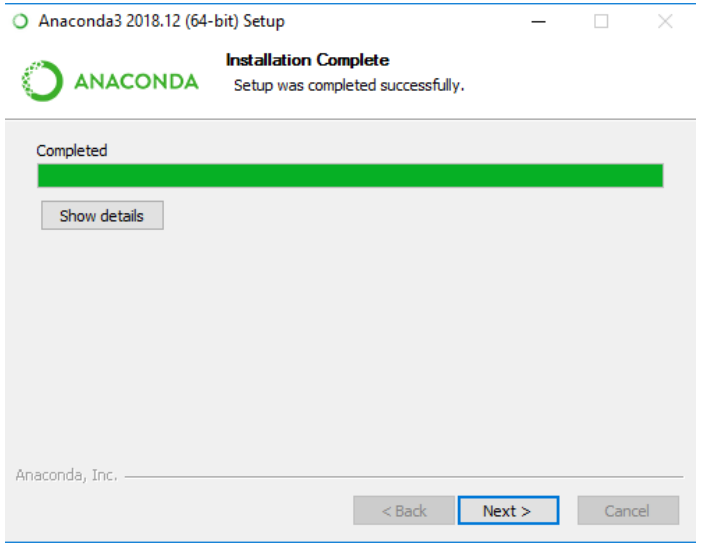


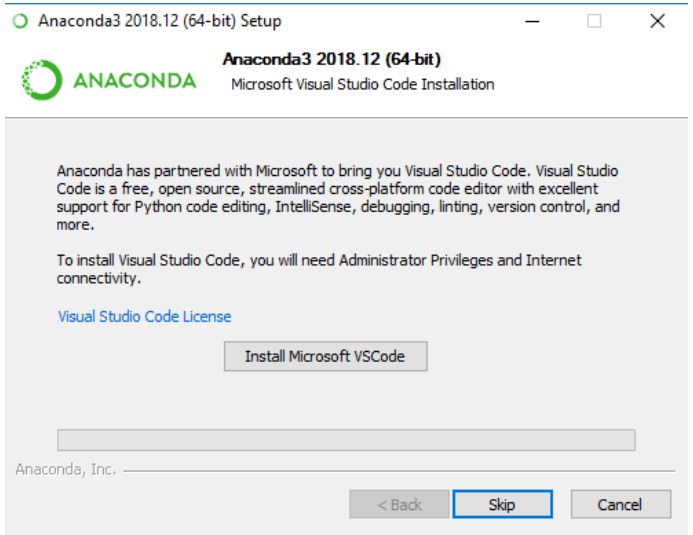
Getting Started With Python Using Anaconda

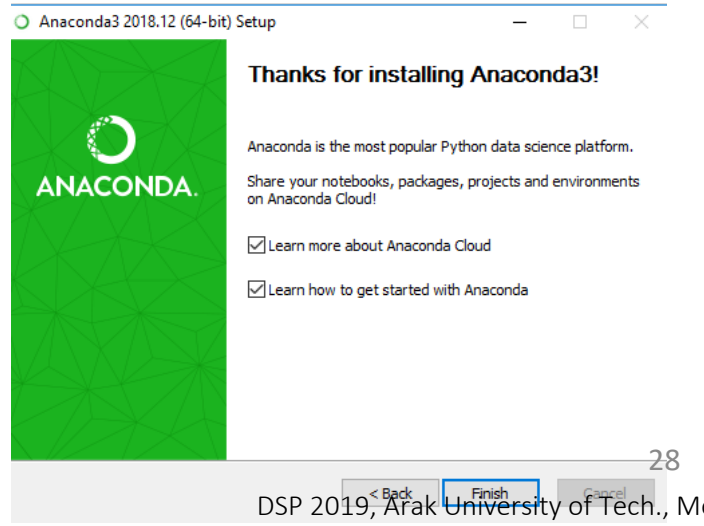




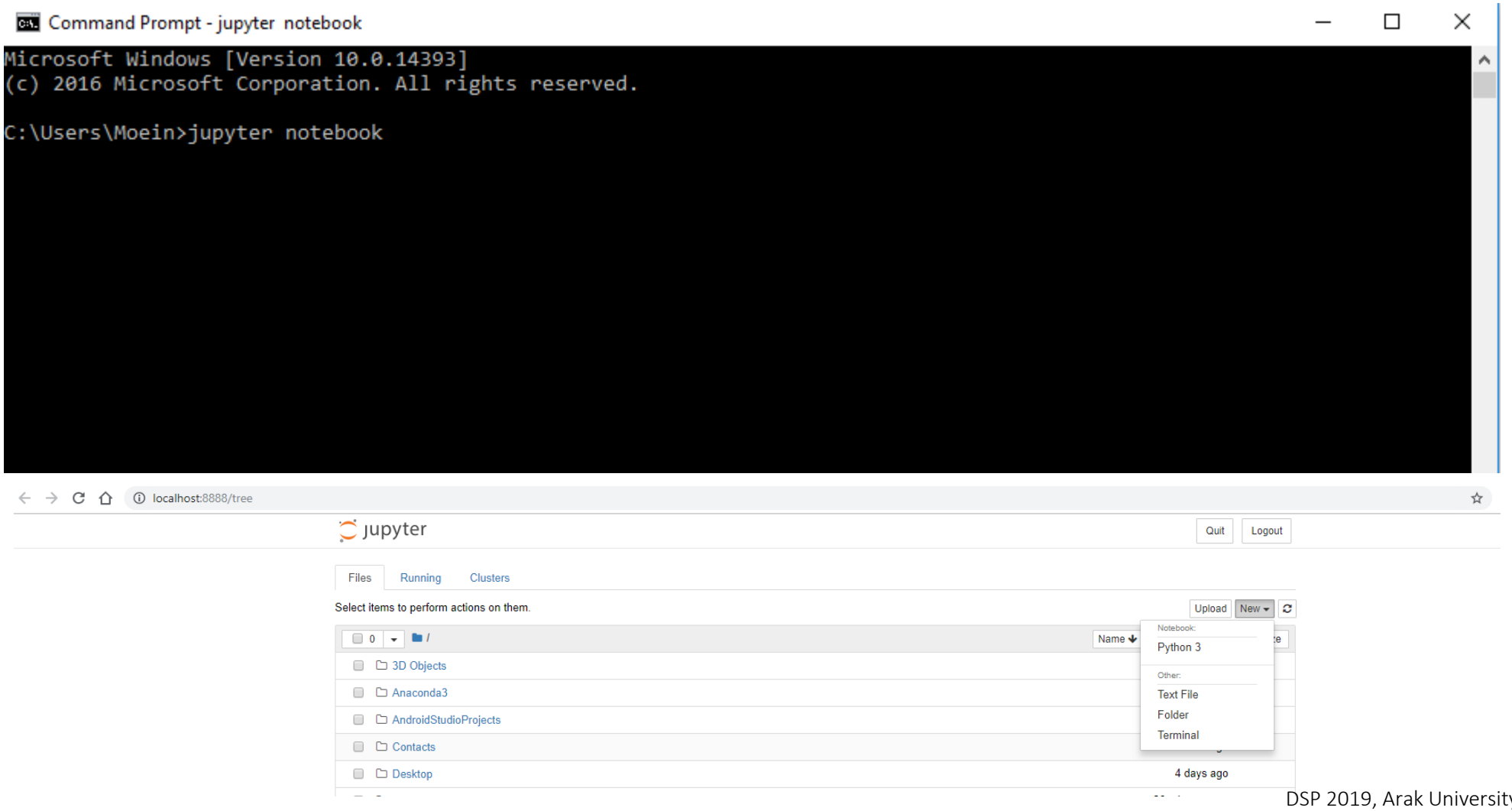




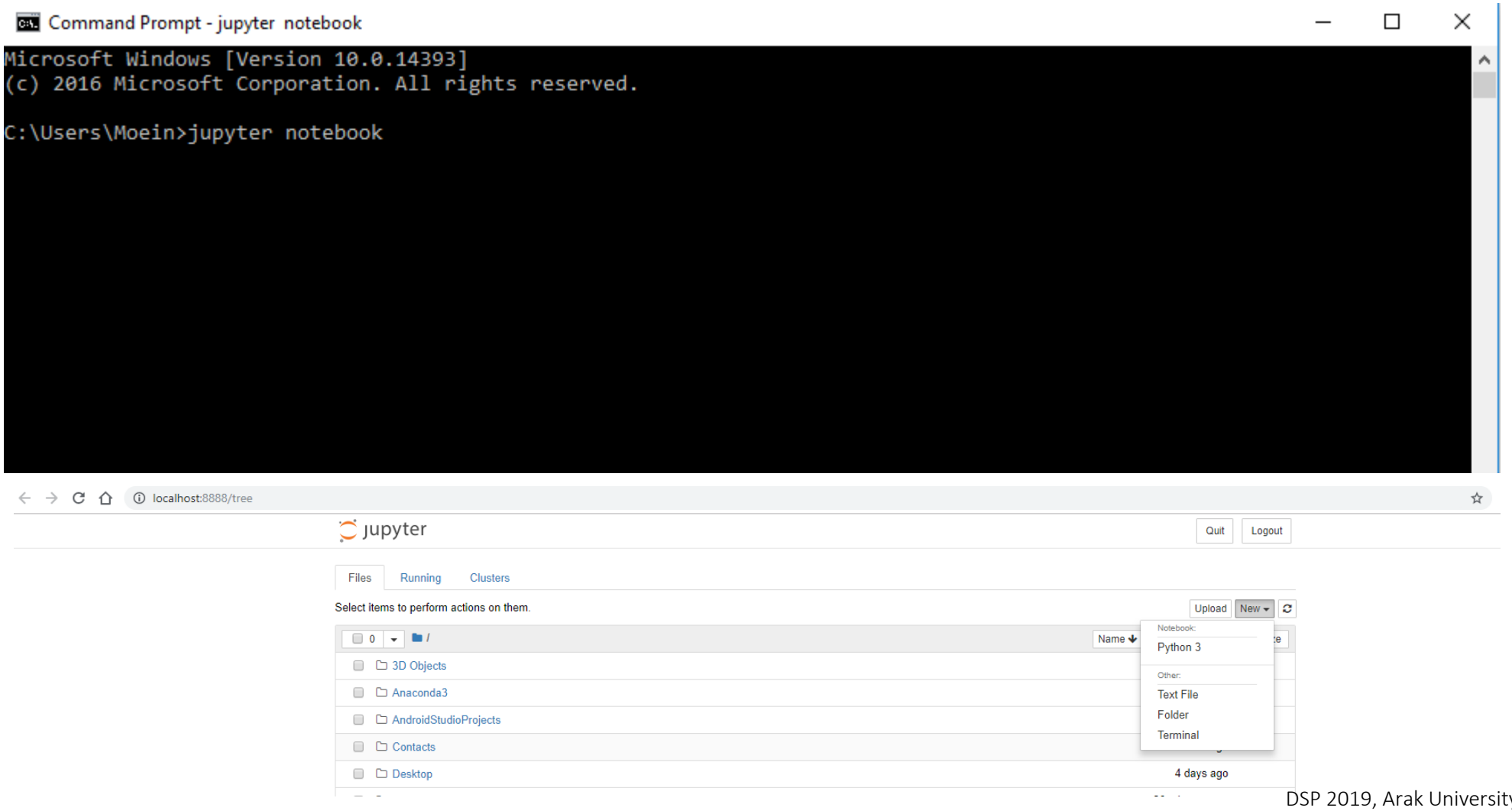




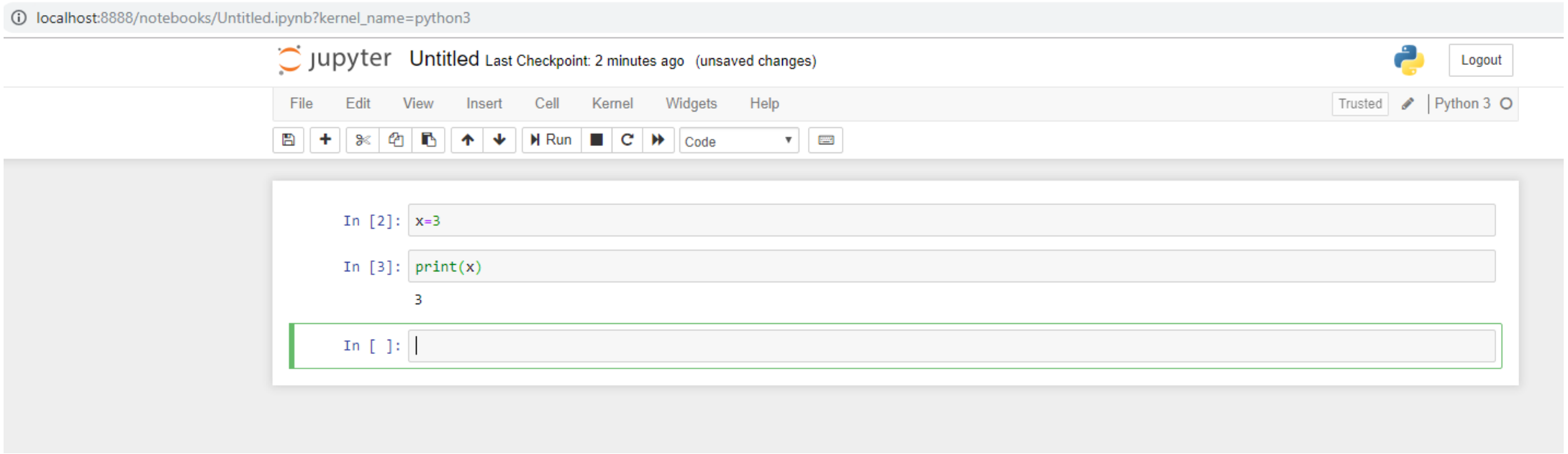
Getting Started With Python Using Anaconda



Getting Started With Python Using Anaconda

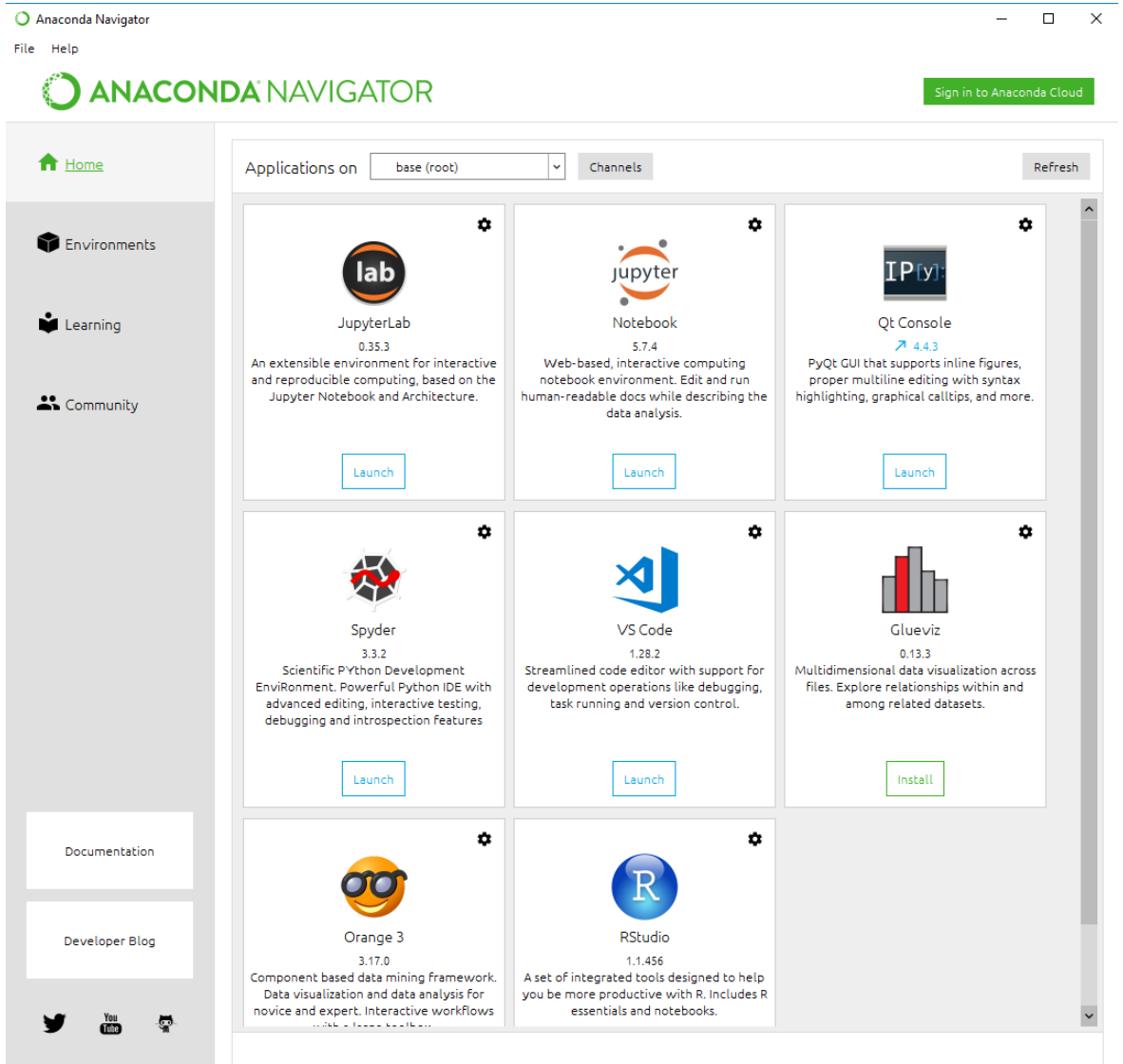


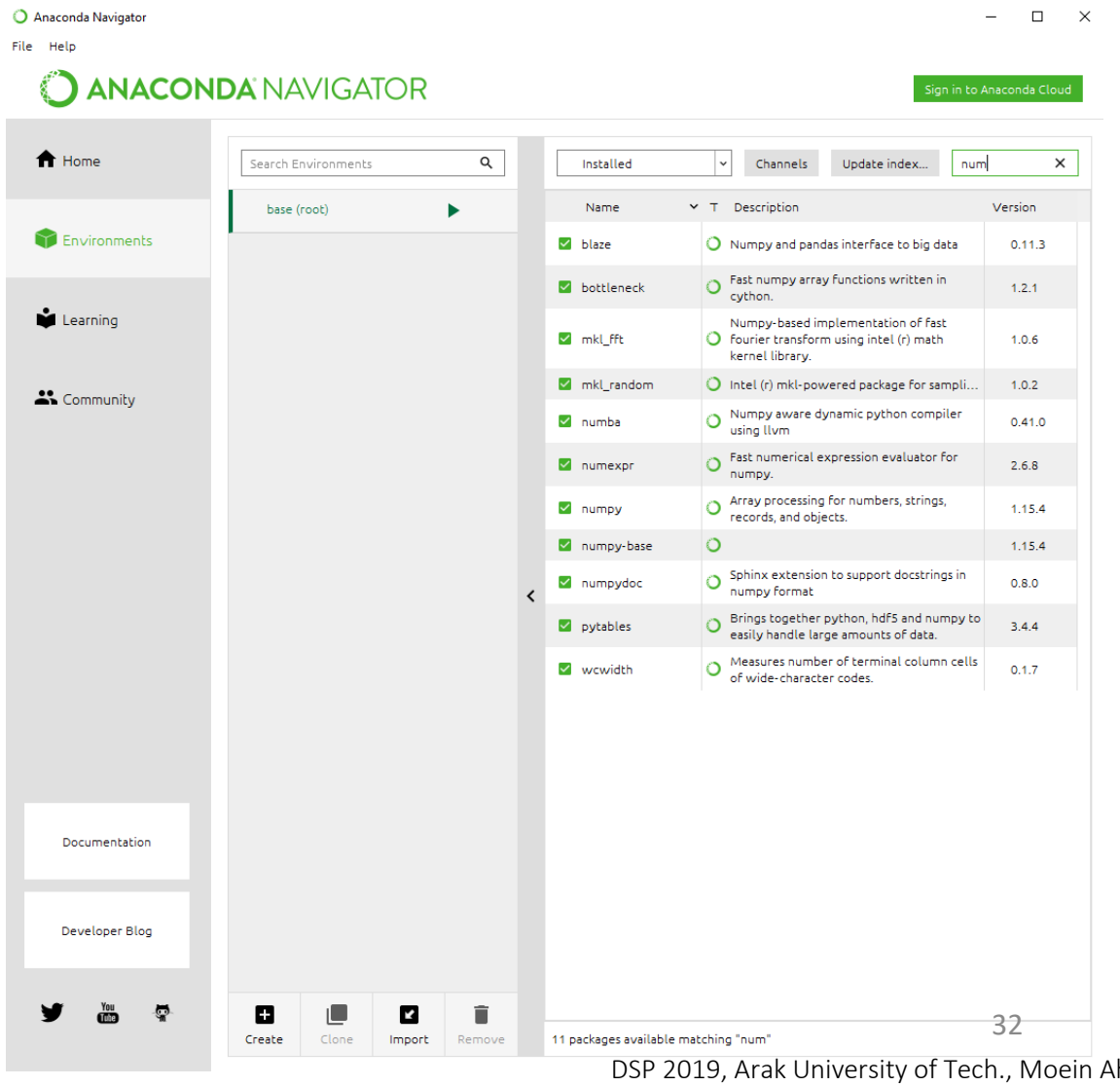
Getting Started With Python Using Anaconda



Run: Shift+Enter

Getting Started With Python Using Anaconda





Welcome to Python!

Python is a high-level programming language, with applications in numerous areas, including web programming, scripting, scientific computing, and artificial intelligence.

It is very popular and used by organizations such as Google, NASA, the CIA, and Disney.

The three major versions of Python are 1.x, 2.x and 3.x. These are subdivided into minor versions, such as 2.7 and 3.3.
Code written for Python 3.x is guaranteed to work in all future versions.
Both Python Version 2.x and 3.x are used currently.
This course covers **Python 3.x**, but it isn't hard to change from one version to another.

First Program

```
>>> print('Hello world!')
Hello world!
```

Simple Operations

```
>>> 2 * (3 + 4)
14
>>> 10 / 2
5.0
```

```
>>> -7
-7
>>> (-7 + 2) * (-4)
20
```

```
>>> 11 / 0
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

Floats

```
>>> 3/4
0.75
>>> 9.8765000
9.8765
```

```
>>> 8 / 2
4.0
>>> 6 * 7.0
42.0
>>> 4 + 1.65
5.65
```

Exponentiation

```
>>> 2**5
32
>>> 9 ** (1/2)
3.0
```

Quotient & Remainder

```
>>> 20 // 6
3
>>> 1.25 % 0.5
0.25
```

Strings

```
>>> "Python is fun!"
'Python is fun!'
>>> 'Always look on the bright side of life'
'Always look on the bright side of life'
```

```
>>> 'Brian\'s mother: He\'s not the Messiah. He\'s a very naughty boy!'
'Brian's mother: He's not the Messiah. He's a very naughty boy!'
```

```
>>> "Spam" + 'eggs'
'Spameggs'
```

```
>>> print("spam" * 3)
spamspamspam

>>> 4 * '2'
'2222'
```

```
>>> "2" + "2"
'22'
>>> 1 + '2' + 3 + '4'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for
 +: 'int' and 'str'
```

Input

```
>>> input("Enter something please: ")
Enter something please: This is what\nthe user enters!

'This is what\\nthe user enters!'
```

Type Conversion

```
>>> "2" + "3"
'23'
>>> int("2") + int("3")
5

>>> float(input("Enter a number: ")) + float(input("Enter
another number: "))
Enter a number: 40
Enter another number: 2
42.0
```

Variables

```
>>> x = 7
>>> print(x)
7
>>> print(x + 3)
10
>>> print(x)
7
```

```
>>> x = 123.456
>>> print(x)
123.456
>>> x = "This is a string"
>>> print(x + "!")
This is a string!
```

```
>>> this_is_a_normal_name = 7

>>> 123abc = 7
SyntaxError: invalid syntax

>>> spaces are not allowed
SyntaxError: invalid syntax
```

```
>>> foo = "a string"
>>> foo
'a string'
>>> bar
NameError: name 'bar' is not defined
>>> del foo
>>> foo
NameError: name 'foo' is not defined
```

Variables

```
>>> foo = input("Enter a number: ")
Enter a number: 7
>>> print(foo)
7
```

In-Place Operators

```
>>> x = 2
>>> print(x)
2
>>> x += 3
>>> print(x)
5
```

```
>>> x = "spam"
>>> print(x)
spam

>>> x += "eggs"
>>> print(x)
spameggs
```

Comparisons: Booleans

```
>>> my_boolean = True
>>> my_boolean
True

>>> 2 == 3
False
>>> "hello" == "hello"
True


>>> 1 != 1
False
>>> "eleven" != "seven"
True
>>> 2 != 10
True
```

```
>>> 7 > 5
True
>>> 10 < 10
False


>>> 7 <= 8
True
>>> 9 >= 9.0
True
```

if Statements

```
if 10 > 5:
  print("10 greater than 5")
```

```
num = 12
if num > 5:
  print("Bigger than 5")
  if num <=47:
    print("Between 5 and 47")
```

```
x = 4
if x == 5:
  print("Yes")
else:
  print("No")
```

```
num = 7
if num == 5:
 print("Number is 5")
else:
 if num == 11:
  print("Number is 11")
 else:
  if num == 7:
   print("Number is 7")
  else:
   print("Number isn't 5, 11 or 7")
```

```
num = 7
if num == 5:
  print("Number is 5")
elif num == 11:
  print("Number is 11")
elif num == 7:
  print("Number is 7")
else:
  print("Number isn't 5, 11 or 7")
```

Boolean Logic

```
>>> 1 == 1 and 2 == 2
True
>>> 1 == 1 and 2 == 3
False
>>> 1 != 1 and 2 == 2
False
>>> 2 < 1 and 3 > 6
False
```

```
>>> 1 == 1 or 2 == 2
True
>>> 1 == 1 or 2 == 3
True
>>> 1 != 1 or 2 == 2
True
>>> 2 < 1 or 3 > 6
False
```

```
>>> not 1 == 1
False
>>> not 1 > 7
True
```

Operator Precedence

Operator	Description
**	Exponentiation (raise to the power)
~, +, -	Complement, unary plus and minus (method names for the last two are +@ and -@)
*, /, %, //	Multiply, divide, modulo and floor division
+, -	Addition and subtraction
>>, <<	Right and left bitwise shift
&	Bitwise 'AND'
^	Bitwise exclusive 'OR'
|	Bitwise 'OR'
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparison operators, equality operators, membership and identity operators
not	Boolean 'NOT'
and	Boolean 'AND'
or	Boolean 'OR'
=, %=, /=, //=, -=, +=, *=, **=	Assignment operators

```
>>> False == False or True
True
>>> False == (False or True)
False
>>> (False == False) or True
True
```

while Loops

```
i = 1
while i <=5:
  print(i)
  i = i + 1

print("Finished!")
```

```
>>>
1
2
3
4
5
Finished!
>>>
```

```
i = 0
while True:
  i = i +1
  if i == 2:
    print("Skipping 2")
    continue
  if i == 5:
    print("Breaking")
    break
  print(i)

print("Finished")
```

Lists

```
words = ["Hello", "world", "!"]
print(words[0])
print(words[1])
print(words[2])
```

```
empty_list = []
print(empty_list)
```

```
number = 3
things = ["string", 0, [1, 2, number], 4.56]
print(things[1])
print(things[2])
print(things[2][2])
```

```
str = "Hello world!"
print(str[6])
```

```
nums = [7, 7, 7, 7, 7]
nums[2] = 5
```

```
nums = [1, 2, 3]
print(nums + [4, 5, 6])
print(nums * 3)
```

```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Lists

```
words = ["spam", "egg", "spam", "sausage"]
print("spam" in words)
print("egg" in words)
print("tomato" in words)
```

```
nums = [1, 2, 3]
print(not 4 in nums)        True
print(4 not in nums)        True
print(not 3 in nums)        False
print(3 not in nums)        False
```

```
nums = [1, 2, 3]
nums.append(4)
```

```
nums = [1, 3, 5, 2, 4]
print(len(nums))
```

```
words = ["Python", "fun"]
index = 1
words.insert(index, "is")
```

```
letters = ['p', 'q', 'r', 's', 'p', 'u']
print(letters.index('r'))
print(letters.index('p'))
print(letters.index('z'))
```

Range

```
numbers = list(range(10))
print(numbers)
```

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```
numbers = list(range(3, 8))
print(numbers)
```

[3, 4, 5, 6, 7]

```
print(range(20) == range(0, 20))
```

True

```
numbers = list(range(5, 20, 2))
print(numbers)
```

[5, 7, 9, 11, 13, 15, 17, 19]

Loops

```
words = ["hello", "world", "spam", "eggs"]
counter = 0
max_index = len(words) - 1

while counter <= max_index:
word = words[counter]
print(word + "!")
counter = counter + 1
```

```
words = ["hello", "world", "spam", "eggs"]
for word in words:
print(word + "!")
```

```
for i in range(5):
print("hello!")
```

Functions

```
def my_func():
  print("spam")
  print("spam")
  print("spam")

my_func()
```

```
def print_with_exclamation(word):
  print(word + "!")

def print_sum_twice(x, y):
  print(x + y)
  print(x + y)
```

```
def function(variable):
  variable += 1
  print(variable)

function(7)
print(variable)
```

```
def max(x, y):
  if x >=y:
    return x
  else:
    return y

print(max(4, 7))
z = max(8, 5)
print(z)
```

```
def add_numbers(x, y):
  total = x + y
  return total
  print("This won't be printed")

print(add_numbers(4, 5))
```

Functions

```
def multiply(x, y):
   return x * y

a = 4
b = 7
operation = multiply
print(operation(a, b))
```

```
def add(x, y):
  return x + y

def do_twice(func, x, y):
  return func(func(x, y), func(x, y))

a = 5
b = 10

print(do_twice(add, a, b))
```

Modules

```
import random

for i in range(5):
  value = random.randint(1, 6)
  print(value)
```

```
from math import pi

print(pi)
```

```
from math import sqrt as square_root
print(square_root(100))
```

```
from math import pi, sqrt
```

Files

```
myfile = open("filename.txt")
```

```
# write mode
open("filename.txt", "w")

# read mode
open("filename.txt", "r")
open("filename.txt")

# binary write mode
open("filename.txt", "wb")
```

```
file = open("filename.txt", "w")
# do stuff to the file
file.close()
```

Files

```
myfile = open("filename.txt")
```

```
# write mode
open("filename.txt", "w")

# read mode
open("filename.txt", "r")
open("filename.txt")

# binary write mode
open("filename.txt", "wb")
```

```
file = open("filename.txt", "w")
# do stuff to the file
file.close()
```

```
file = open("filename.txt", "r")
cont = file.read()
print(cont)
file.close()
```

```
file = open("filename.txt", "r")
print(file.read(16))
print(file.read(4))
print(file.read(4))
print(file.read())
file.close()
```

```
file = open("filename.txt", "r")
print(file.readlines())
file.close()
```

```
file = open("filename.txt", "r")
for line in file:
    print(line)
file.close()
```

Files

```
file = open("newfile.txt", "w")
file.write("This has been written to a file")
file.close()

file = open("newfile.txt", "r")
print(file.read())
file.close()
```

```
msg = "Hello world!"
file = open("newfile.txt", "w")
amount_written = file.write(msg)
print(amount_written)
file.close()
```

```
with open("filename.txt") as f:
  print(f.read())
```

None

```
>>> None == None
True
>>> None
>>> print(None)
None
```

```
def some_func():
  print("Hi!")

var = some_func()
print(var)
```

```
Hi!
None
```

Dictionaries

```
ages = {"Dave": 24, "Mary": 42, "John": 58}
print(ages["Dave"])
print(ages["Mary"])
```

```
primary = {
  "red": [255, 0, 0],
  "green": [0, 255, 0],
  "blue": [0, 0, 255],
}

print(primary["red"])
print(primary["yellow"])
```

Dictionaries

```
squares = {1: 1, 2: 4, 3: "error", 4: 16,}
squares[8] = 64
squares[3] = 9
print(squares)
```

```
{8: 64, 1: 1, 2: 4, 3: 9, 4: 16}
```

```
pairs = {1: "apple",
 "orange": [2, 3, 4],
 True: False,
 None: "True",
}

print(pairs.get("orange"))
print(pairs.get(7))
print(pairs.get(12345, "not in dictionary"))
```

```
[2, 3, 4]
None
not in dictionary
```

Tuples

```
words = ("spam", "eggs", "sausages",)

 my_tuple = "one", "two", "three"
```

List Slices

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
print(squares[2:6])
print(squares[3:8])
print(squares[0:1])
```

```
[4, 9, 16, 25]
[9, 16, 25, 36, 49]
[0]
```

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
print(squares[:7])
print(squares[7:])
```

```
[0, 1, 4, 9, 16, 25, 36]
[49, 64, 81]
```

List Slices

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
print(squares[::2])
print(squares[2:8:3])
```

```
[0, 4, 16, 36, 64]
[4, 25]
```

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
print(squares[1:-1])
```

```
[1, 4, 9, 16, 25, 36, 49, 64]
```

List Comprehensions

```
cubes = [i**3 for i in range(5)]
```

```
[0, 1, 8, 27, 64]
```

```
evens=[i**2 for i in range(10) if i**2 % 2 == 0]
```

```
[0, 4, 16, 36, 64]
```

String Formatting

```
nums = [4, 5, 6]
msg = "Numbers: {0} {1} {2}". format(nums[0], nums[1],
nums[2])
print(msg)
```

Numbers: 4 5 6

```
a = "{x}, {y}".format(x=5, y=12)
print(a)
```

5, 12

String Functions

```
print(", ".join(["spam", "eggs", "ham"]))
#prints "spam, eggs, ham"

print("Hello ME".replace("ME", "world"))
#prints "Hello world"

print("This is a sentence.".startswith("This"))
# prints "True"

print("This is a sentence.".endswith("sentence."))
# prints "True"

print("This is a sentence.".upper())
# prints "THIS IS A SENTENCE."

print("AN ALL CAPS SENTENCE".lower())
#prints "an all caps sentence"

print("spam, eggs, ham".split(", "))
#prints "['spam', 'eggs', 'ham']"
```

C++

C++ is a general-purpose programming language.
C++ is used to create computer programs. Anything from art applications, music players and even video games!

```
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello world!";
  return 0;
}
```

```
int myVariable = 10;


int mark = 90;

if (mark < 50) {
cout << "You failed." << endl;
}
else {
cout << "You passed." << endl;
}
```

C++ Qt

main.cpp

```cpp
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}

```

mainwindow.cpp — MainWindow::~MainWindow()

```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    qDebug()<<"Hello world!";;
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

mainwindow.h — <Select Symbol>

```cpp
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QDebug>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

python™ 110 msec

```python
import time
a = time.clock()
prim =[]
n = 2
while True:
    isprime = True
    for p in prim:
        if n%p==0:
            isprime=False
            break
    if isprime:
        prim.append(n)
    n +=1
    if len(prim)>=1000:
        break
sum = 0
for s in prim:
    sum+=s
b = time.clock()
print((b-a)*1000)
print(sum)
```

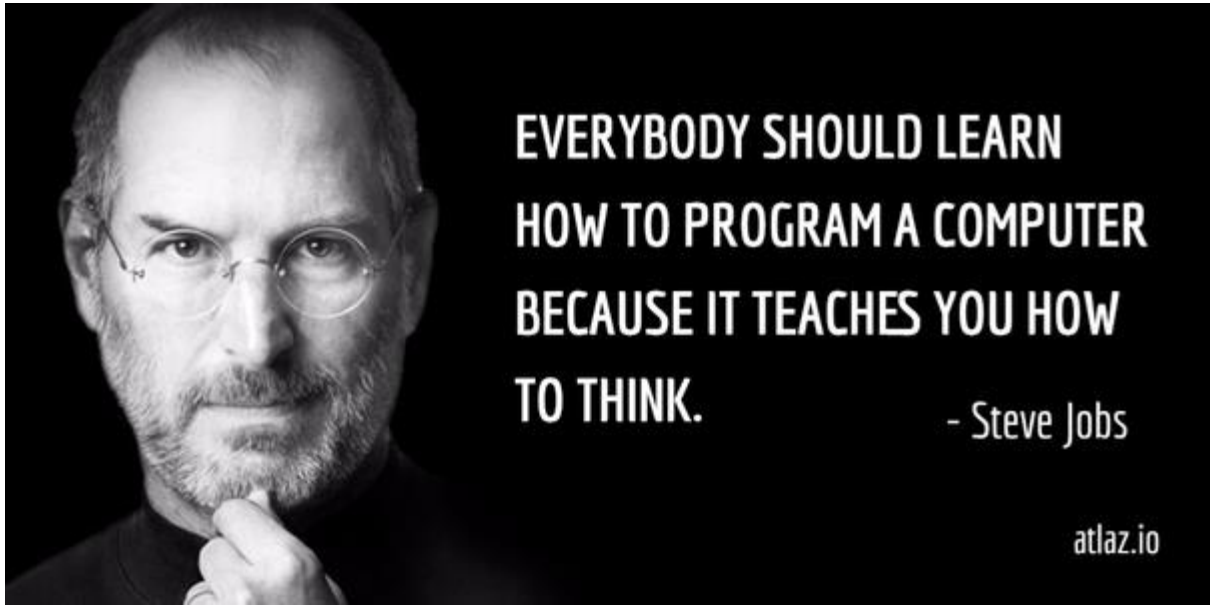
```
108.813108677
3682913
```

C++ Qt 3 msec

```
11      QElapsedTimer time;
12      time.restart();
13      QVector<int> prim;
14      int n = 2;
15      bool isprime;
16      while(true){
17          isprime = true;
18          for (int i = 0; i < prim.length(); ++i) {
19              if(n % prim.at(i) == 0){
20                  isprime=false;
21                  break;
22              }
23          }
24          if(isprime)
25              prim.append(n);
26          n++;
27          if (prim.length()>=1000)
28              break;
29      }
30      int sum = 0;
31      for (int i = 0; i < prim.length(); ++i) {
32          sum+=prim.at(i);
33      }
34      qDebug()<<time.elapsed();
35      qDebug()<<sum;
36
37  }
```

```
Application Output
ArakUT
Starting E:\Roshd\Qt\ArakUT\release\ArakUT.exe...
3
3682913
```


CHOOSE YOUR WEAPON
C++
JAVA
PYTHON
C


EVERYBODY SHOULD LEARN
HOW TO PROGRAM A COMPUTER
BECAUSE IT TEACHES YOU HOW
TO THINK.
- Steve Jobs
atlaz.io

mit media lab

Massachusetts Institute of Technology

Mitchel Resnick

I'm the LEGO Papert Professor of Learning Research at the MIT Media Lab,
where I lead the Lifelong Kindergarten research group.

هر کسی که در قرن بیست و یکم برنامه‌نویسی بلد نباشد، بی‌سواد است.

python™

Basic data types:

Numbers, Booleans, Strings

Numbers: Integers and floats work as you would expect from other languages:

```
x = 3
print(type(x)) # Prints "<class 'int'>"
print(x)       # Prints "3"
print(x + 1)   # Addition; prints "4"
print(x - 1)   # Subtraction; prints "2"
print(x * 2)   # Multiplication; prints "6"
print(x ** 2)  # Exponentiation; prints "9"
x += 1
print(x)  # Prints "4"
x *= 2
print(x)  # Prints "8"
y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # Prints "2.5 3.5 5.0 6.25"
```

Booleans: Python implements all of the usual operators for Boolean logic, but uses English words rather than symbols (&&, ||, etc.):

```
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"
print(t or f)  # Logical OR; prints "True"
print(not t)   # Logical NOT; prints "False"
print(t != f)  # Logical XOR; prints "True"
```

python™

Basic data types:

Numbers, Booleans, Strings

Strings: Python has great support for strings:

```
hello = 'hello'    # String literals can use single quotes
world = "world"    # or double quotes; it does not matter.
print(hello)       # Prints "hello"
print(len(hello))  # String length; prints "5"
hw = hello + ' ' + world  # String concatenation
print(hw)  # prints "hello world"
hw12 = '%s %s %d' % (hello, world, 12)  # sprintf style string formatting
print(hw12)  # prints "hello world 12"
```

```
s = "hello"
print(s.capitalize())  # Capitalize a string; prints "Hello"
print(s.upper())       # Convert a string to uppercase; prints "HELLO"
print(s.rjust(7))      # Right-justify a string, padding with spaces; prints "  hello"
print(s.center(7))     # Center a string, padding with spaces; prints " hello "
print(s.replace('l', '(ell)'))  # Replace all instances of one substring with another;
                                # prints "he(ell)(ell)o"
print('  world '.strip())  # Strip leading and trailing whitespace; prints "world"
```

python™

Containers: List, Dictionary, Set, Tuple

List

```
xs = [3, 1, 2]    # Create a list
print(xs, xs[2])  # Prints "[3, 1, 2] 2"
print(xs[-1])     # Negative indices count from the end of the list; prints "2"
xs[2] = 'foo'     # Lists can contain elements of different types
print(xs)         # Prints "[3, 1, 'foo']"
xs.append('bar')  # Add a new element to the end of the list
print(xs)         # Prints "[3, 1, 'foo', 'bar']"
x = xs.pop()      # Remove and return the last element of the list
print(x, xs)      # Prints "bar [3, 1, 'foo']"
```

```
nums = list(range(5))     # range is a built-in function that creates a list of integers
print(nums)               # Prints "[0, 1, 2, 3, 4]"
print(nums[2:4])          # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print(nums[2:])           # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print(nums[:2])           # Get a slice from the start to index 2 (exclusive); prints "[0, 1]
print(nums[:])            # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
print(nums[:-1])          # Slice indices can be negative; prints "[0, 1, 2, 3]"
nums[2:4] = [8, 9]        # Assign a new sublist to a slice
print(nums)               # Prints "[0, 1, 8, 9, 4]"
```

Containers: List, Dictionary, Set, Tuple

List

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
# Prints "cat", "dog", "monkey", each on its own line.
```

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# Prints "#1: cat", "#2: dog", "#3: monkey", each on its own line
```

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)   # Prints [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print(squares)   # Prints [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)  # Prints "[0, 4, 16]"
```

python™

Containers: List, Dictionary, Set, Tuple

Dictionary

```
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet'     # Set an entry in a dictionary
print(d['fish'])      # Prints "wet"
# print(d['monkey'])  # KeyError: 'monkey' not a key of d
print(d.get('monkey', 'N/A'))  # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
del d['fish']         # Remove an element from a dictionary
print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"
```

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
```

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
```

```
nums = [0, 1, 2, 3, 4]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print(even_num_to_square)  # Prints "{0: 0, 2: 4, 4: 16}"
```

python™

Containers: List, Dictionary, Set, Tuple

Set

```
animals = {'cat', 'dog'}
print('cat' in animals)   # Check if an element is in a set; prints "True"
print('fish' in animals)  # prints "False"
animals.add('fish')       # Add an element to a set
print('fish' in animals)  # Prints "True"
print(len(animals))       # Number of elements in a set; prints "3"
animals.add('cat')        # Adding an element that is already in the set does nothing
print(len(animals))       # Prints "3"
animals.remove('cat')     # Remove an element from a set
print(len(animals))       # Prints "2"
```

```
animals = {'cat', 'dog', 'fish'}
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# Prints "#1: fish", "#2: dog", "#3: cat"
```

```
from math import sqrt
nums = {int(sqrt(x)) for x in range(30)}
print(nums)  # Prints "{0, 1, 2, 3, 4, 5}"
```

Containers: List, Dictionary, Set, Tuple

Tuple

tuple is in many ways similar to a list; one of the most important differences is that tuples can be used as keys in dictionaries and as elements of sets, while lists cannot.

```
d = {(x, x + 1): x for x in range(10)}  # Create a dictionary with tuple keys
t = (5, 6)        # Create a tuple
print(type(t))    # Prints "<class 'tuple'>"
print(d[t])       # Prints "5"
print(d[(1, 2)])  # Prints "1"
```

python™

Functions

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
# Prints "negative", "zero", "positive"
```

```
def hello(name, loud=False):
    if loud:
        print('HELLO, %s!' % name.upper())
    else:
        print('Hello, %s' % name)

hello('Bob') # Prints "Hello, Bob"
hello('Fred', loud=True)  # Prints "HELLO, FRED!"
```

Classes

```
class Greeter(object):

    # Constructor
    def __init__(self, name):
        self.name = name  # Create an instance variable

    # Instance method
    def greet(self, loud=False):
        if loud:
            print('HELLO, %s!' % self.name.upper())
        else:
            print('Hello, %s' % self.name)

g = Greeter('Fred')  # Construct an instance of the Greeter class
g.greet()            # Call an instance method; prints "Hello, Fred"
g.greet(loud=True)   # Call an instance method; prints "HELLO, FRED!"
```

python™

libraries Numpy SciPy Matplotlib

Numpy Arrays

```
import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

```
import numpy as np

a = np.zeros((2,2))   # Create an array of all zeros
print(a)              # Prints "[[ 0.  0.]
                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones
print(b)              # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7)  # Create a constant array
print(c)               # Prints "[[ 7.  7.]
                       #          [ 7.  7.]]"

d = np.eye(2)         # Create a 2x2 identity matrix
print(d)              # Prints "[[ 1.  0.]
                      #          [ 0.  1.]]"

e = np.random.random((2,2))  # Create an array filled with random values
print(e)                     # Might print "[[ 0.91940167  0.08143941]
                             #              [ 0.68744134  0.87236687]]"
```

python™

libraries Numpy SciPy Matplotlib

Numpy Array indexing

```python
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
print(a[0, 1])   # Prints "2"
b[0, 0] = 77     # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1])   # Prints "77"
```

```python
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Two ways of accessing the data in the middle row of the array.
# Mixing integer indexing with slices yields an array of lower rank,
# while using only slices yields an array of the same rank as the
# original array:
row_r1 = a[1, :]    # Rank 1 view of the second row of a
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
print(row_r1, row_r1.shape)  # Prints "[5 6 7 8] (4,)"
print(row_r2, row_r2.shape)  # Prints "[[5 6 7 8]] (1, 4)"

# We can make the same distinction when accessing columns of an array:
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print(col_r1, col_r1.shape)  # Prints "[ 2  6 10] (3,)"
print(col_r2, col_r2.shape)  # Prints "[[ 2]
                             #         [ 6]
                             #         [10]] (3, 1)"
```

python™

libraries Numpy SciPy Matplotlib

Numpy Array indexing

```
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

# An example of integer array indexing.
# The returned array will have shape (3,) and
print(a[[0, 1, 2], [0, 1, 0]])  # Prints "[1 4 5]"

# The above example of integer array indexing is equivalent to this:
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))  # Prints "[1 4 5]"

# When using integer array indexing, you can reuse the same
# element from the source array:
print(a[[0, 0], [1, 1]])  # Prints "[2 2]"

# Equivalent to the previous integer array indexing example
print(np.array([a[0, 1], a[0, 1]]))  # Prints "[2 2]"
```

```
import numpy as np

# Create a new array from which we will select elements
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])

print(a)  # prints "array([[ 1,  2,  3],
          #                [ 4,  5,  6],
          #                [ 7,  8,  9],
          #                [10, 11, 12]])"

# Create an array of indices
b = np.array([0, 2, 0, 1])

# Select one element from each row of a using the indices in b
print(a[np.arange(4), b])  # Prints "[ 1  6  7 11]"

# Mutate one element from each row of a using the indices in b
a[np.arange(4), b] += 10

print(a)  # prints "array([[11,  2,  3],
          #                [ 4,  5, 16],
          #                [17,  8,  9],
          #                [10, 21, 12]])
```

python™

libraries Numpy SciPy Matplotlib

Numpy Array indexing

```
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

bool_idx = (a > 2)   # Find the elements of a that are bigger than 2;
                     # this returns a numpy array of Booleans of the same
                     # shape as a, where each slot of bool_idx tells
                     # whether that element of a is > 2.

print(bool_idx)      # Prints "[[False False]
                     #          [ True  True]
                     #          [ True  True]]"

# We use boolean array indexing to construct a rank 1 array
# consisting of the elements of a corresponding to the True values
# of bool_idx
print(a[bool_idx])  # Prints "[3 4 5 6]"

# We can do all of the above in a single concise statement:
print(a[a > 2])     # Prints "[3 4 5 6]"
```

libraries Numpy SciPy Matplotlib

Numpy Datatypes

```
import numpy as np

x = np.array([1, 2])   # Let numpy choose the datatype
print(x.dtype)         # Prints "int64"

x = np.array([1.0, 2.0])   # Let numpy choose the datatype
print(x.dtype)             # Prints "float64"

x = np.array([1, 2], dtype=np.int64)   # Force a particular datatype
print(x.dtype)                         # Prints "int64"
```

libraries Numpy SciPy Matplotlib

Numpy Array math

```
import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print(v.dot(w))
print(np.dot(v, w))

# Matrix / vector product; both produce the rank 1 array [29 67]
print(x.dot(v))
print(np.dot(x, v))

# Matrix / matrix product; both produce the rank 2 array
# [[19 22]
#  [43 50]]
print(x.dot(y))
print(np.dot(x, y))
```

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))

# Elementwise difference; both produce the array
# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print(x - y)
print(np.subtract(x, y))

# Elementwise product; both produce the array
# [[ 5.0 12.0]
#  [21.0 32.0]]
print(x * y)
print(np.multiply(x, y))

# Elementwise division; both produce the array
# [[ 0.2         0.33333333]
#  [ 0.42857143  0.5       ]]
print(x / y)
print(np.divide(x, y))

# Elementwise square root; produces the array
# [[ 1.          1.41421356]
#  [ 1.73205081  2.        ]]
print(np.sqrt(x))
```

python™

libraries Numpy SciPy Matplotlib

Numpy Array math

```
import numpy as np

x = np.array([[1,2],[3,4]])

print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"
```

```
import numpy as np

x = np.array([[1,2], [3,4]])
print(x)    # Prints "[[1 2]
            #         [3 4]]"
print(x.T)  # Prints "[[1 3]
            #         [2 4]]"

# Note that taking the transpose of a rank 1 array does nothing:
v = np.array([1,2,3])
print(v)    # Prints "[1 2 3]"
print(v.T)  # Prints "[1 2 3]"
```

libraries Numpy SciPy Matplotlib

Numpy Broadcasting

```python
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = np.empty_like(x)   # Create an empty matrix with the same shape as x

# Add the vector v to each row of the matrix x with an explicit loop
for i in range(4):
    y[i, :] = x[i, :] + v

# Now y is the following
# [[ 2  2  4]
#  [ 5  5  7]
#  [ 8  8 10]
#  [11 11 13]]
print(y)
```

```python
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))   # Stack 4 copies of v on top of each other
print(vv)                 # Prints "[[1 0 1]
                          #         [1 0 1]
                          #         [1 0 1]
                          #         [1 0 1]]"
y = x + vv  # Add x and vv elementwise
print(y)  # Prints "[[ 2  2  4
          #         [ 5  5  7]
          #         [ 8  8 10]
          #         [11 11 13]]"
```

python™

libraries Numpy SciPy Matplotlib

Numpy Broadcasting

```
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v  # Add v to each row of x using broadcasting
print(y)  # Prints "[[ 2  2  4]
          #         [ 5  5  7]
          #         [ 8  8 10]
          #         [11 11 13]]"
```

```python
import numpy as np

# Compute outer product of vectors
v = np.array([1,2,3])  # v has shape (3,)
w = np.array([4,5])    # w has shape (2,)
# To compute an outer product, we first reshape v to be a column
# vector of shape (3, 1); we can then broadcast it against w to yield
# an output of shape (3, 2), which is the outer product of v and w:
# [[ 4  5]
#  [ 8 10]
#  [12 15]]
print(np.reshape(v, (3, 1)) * w)

# Add a vector to each row of a matrix
x = np.array([[1,2,3], [4,5,6]])
# x has shape (2, 3) and v has shape (3,) so they broadcast to (2, 3),
# giving the following matrix:
# [[2 4 6]
#  [5 7 9]]
print(x + v)

# Add a vector to each column of a matrix
# x has shape (2, 3) and w has shape (2,).
# If we transpose x then it has shape (3, 2) and can be broadcast
# against w to yield a result of shape (3, 2); transposing this result
# yields the final result of shape (2, 3) which is the matrix x with
# the vector w added to each column. Gives the following matrix:
# [[ 5  6  7]
#  [ 9 10 11]]
print((x.T + w).T)
# Another solution is to reshape w to be a column vector of shape (2, 1);
# we can then broadcast it directly against x to produce the same
# output.
print(x + np.reshape(w, (2, 1)))

# Multiply a matrix by a constant:
# x has shape (2, 3). Numpy treats scalars as arrays of shape ();
# these can be broadcast together to shape (2, 3), producing the
# following array:
# [[ 2  4  6]
#  [ 8 10 12]]
print(x * 2)
```

python™

libraries Numpy SciPy Matplotlib

SciPy

```python
from scipy.misc import imread, imsave, imresize

# Read an JPEG image into a numpy array
img = imread('assets/cat.jpg')
print(img.dtype, img.shape)  # Prints "uint8 (400, 248, 3)"

# We can tint the image by scaling each of the color channels
# by a different scalar constant. The image has shape (400, 248, 3);
# we multiply it by the array [1, 0.95, 0.9] of shape (3,);
# numpy broadcasting means that this leaves the red channel unchanged,
# and multiplies the green and blue channels by 0.95 and 0.9
# respectively.
img_tinted = img * [1, 0.95, 0.9]

# Resize the tinted image to be 300 by 300 pixels.
img_tinted = imresize(img_tinted, (300, 300))

# Write the tinted image back to disk
imsave('assets/cat_tinted.jpg', img_tinted)
```

```python
import numpy as np
from scipy.spatial.distance import pdist, squareform

# Create the following array where each row is a point in 2D space:
# [[0 1]
#  [1 0]
#  [2 0]]
x = np.array([[0, 1], [1, 0], [2, 0]])
print(x)

# Compute the Euclidean distance between all rows of x.
# d[i, j] is the Euclidean distance between x[i, :] and x[j, :],
# and d is the following array:
# [[ 0.          1.41421356  2.23606798]
#  [ 1.41421356  0.          1.        ]
#  [ 2.23606798  1.          0.        ]]
d = squareform(pdist(x, 'euclidean'))
print(d)
```

python™

libraries Numpy SciPy Matplotlib

Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# Plot the points using matplotlib
plt.plot(x, y)
plt.show()  # You must call plt.show() to make graphics appear.
```

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

python™

libraries Numpy SciPy Matplotlib

Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Set up a subplot grid that has height 2 and width 1,
# and set the first such subplot as active.
plt.subplot(2, 1, 1)

# Make the first plot
plt.plot(x, y_sin)
plt.title('Sine')

# Set the second subplot as active, and make the second plot.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')

# Show the figure.
plt.show()
```

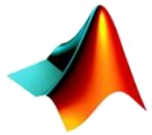
python™

libraries Numpy SciPy Matplotlib

Matplotlib

```
import numpy as np
from scipy.misc import imread, imresize
import matplotlib.pyplot as plt

img = imread('assets/cat.jpg')
img_tinted = img * [1, 0.95, 0.9]

# Show the original image
plt.subplot(1, 2, 1)
plt.imshow(img)

# Show the tinted image
plt.subplot(1, 2, 2)

# A slight gotcha with imshow is that it might give strange results
# if presented with data that is not uint8. To work around this, we
# explicitly cast the image to uint8 before displaying it.
plt.imshow(np.uint8(img_tinted))
plt.show()
```

MATLAB	numpy
`help func`	`info(func)` or `help(func)` or `func?` (in Ipython)
`which func`	see note HELP
`type func`	`source(func)` or `func??` (in Ipython)
`a && b`	`a and b`
`a || b`	`a or b`
`1*i`, `1*j`, `1i`, `1j`	`1j`
`eps`	`np.spacing(1)`
`ode45`	`scipy.integrate.solve_ivp(f)`
`ode15s`	`scipy.integrate.solve_ivp(f, method='BDF')`



MATLAB

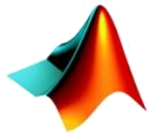
MATLAB	NumPy
`ndims(a)`	`ndim(a)` or `a.ndim`
`numel(a)`	`size(a)` or `a.size`
`size(a)`	`shape(a)` or `a.shape`
`size(a,n)`	`a.shape[n-1]`
`[ 1 2 3; 4 5 6 ]`	`array([[1.,2.,3.], [4.,5.,6.]])`
`[ a b; c d ]`	`block([[a,b], [c,d]])`
`a(end)`	`a[-1]`
`a(2,5)`	`a[1,4]`
`a(2,:)`	`a[1]` or `a[1,:]`
`a(1:5,:)`	`a[0:5]` or `a[:5]` or `a[0:5,:]`
`a(end-4:end,:)`	`a[-5:]`
`a(1:3,5:9)`	`a[0:3][:,4:9]`
`a([2,4,5],[1,3])`	`a[ix_([1,3,4],[0,2])]`

MATLAB	python
`a(3:2:21,:)`	`a[ 2:21:2,:]`
`a(1:2:end,:)`	`a[ ::2,:]`
`a(end:-1:1,:)` or `flipud(a)`	`a[ ::-1,:]`
`a([1:end 1],:)`	`a[r_[:len(a),0]]`
`a.'`	`a.transpose()` or `a.T`
`a'`	`a.conj().transpose()` or `a.conj().T`
`a * b`	`a @ b`
`a .* b`	`a * b`
`a./b`	`a/b`
`a.^3`	`a**3`
`(a>0.5)`	`(a>0.5)`
`find(a>0.5)`	`nonzero(a>0.5)`
`a(:,find(v>0.5))`	`a[:,nonzero(v>0.5)[0]]`
`a(:,find(v>0.5))`	`a[:,v.T>0.5]`
`a(a<0.5)=0`	`a[a<0.5]=0`

MATLAB	python
`a .* (a>0.5)`	`a * (a>0.5)`
`a(:) = 3`	`a[:] = 3`
`y=x`	`y = x.copy()`
`y=x(2,:)`	`y = x[1,:].copy()`
`y=x(:)`	`y = x.flatten()`
`1:10`	`arange(1.,11.)` or `r_[1.:11.]` or `r_[1:10:10j]`
`0:9`	`arange(10.)` or `r_[:10.]` or `r_[:9:10j]`
`[1:10]'`	`arange(1.,11.)[:, newaxis]`
`zeros(3,4)`	`zeros((3,4))`
`zeros(3,4,5)`	`zeros((3,4,5))`
`ones(3,4)`	`ones((3,4))`
`eye(3)`	`eye(3)`
`diag(a)`	`diag(a)`
`diag(a,0)`	`diag(a,0)`

MATLAB	python
`rand(3,4)`	`random.rand(3,4)`
`linspace(1,3,4)`	`linspace(1,3,4)`
`[x,y]=meshgrid(0:8,0:5)`	`mgrid[0:9.,0:6.]` or `meshgrid(r_[0:9.],r_[0:6.]`
	`ogrid[0:9.,0:6.]` or `ix_(r_[0:9.],r_[0:6.]`
`[x,y]=meshgrid([1,2,4],[2,4,5])`	`meshgrid([1,2,4],[2,4,5])`
	`ix_([1,2,4],[2,4,5])`
`repmat(a, m, n)`	`tile(a, (m, n))`
`[a b]`	`concatenate((a,b),1)` or `hstack((a,b))` or `column_stack((a,b))` or `c_[a,b]`
`[a; b]`	`concatenate((a,b))` or `vstack((a,b))` or `r_[a,b]`
`max(max(a))`	`a.max()`
`max(a)`	`a.max(0)`
`max(a,[],2)`	`a.max(1)`
`max(a,b)`	`maximum(a, b)`

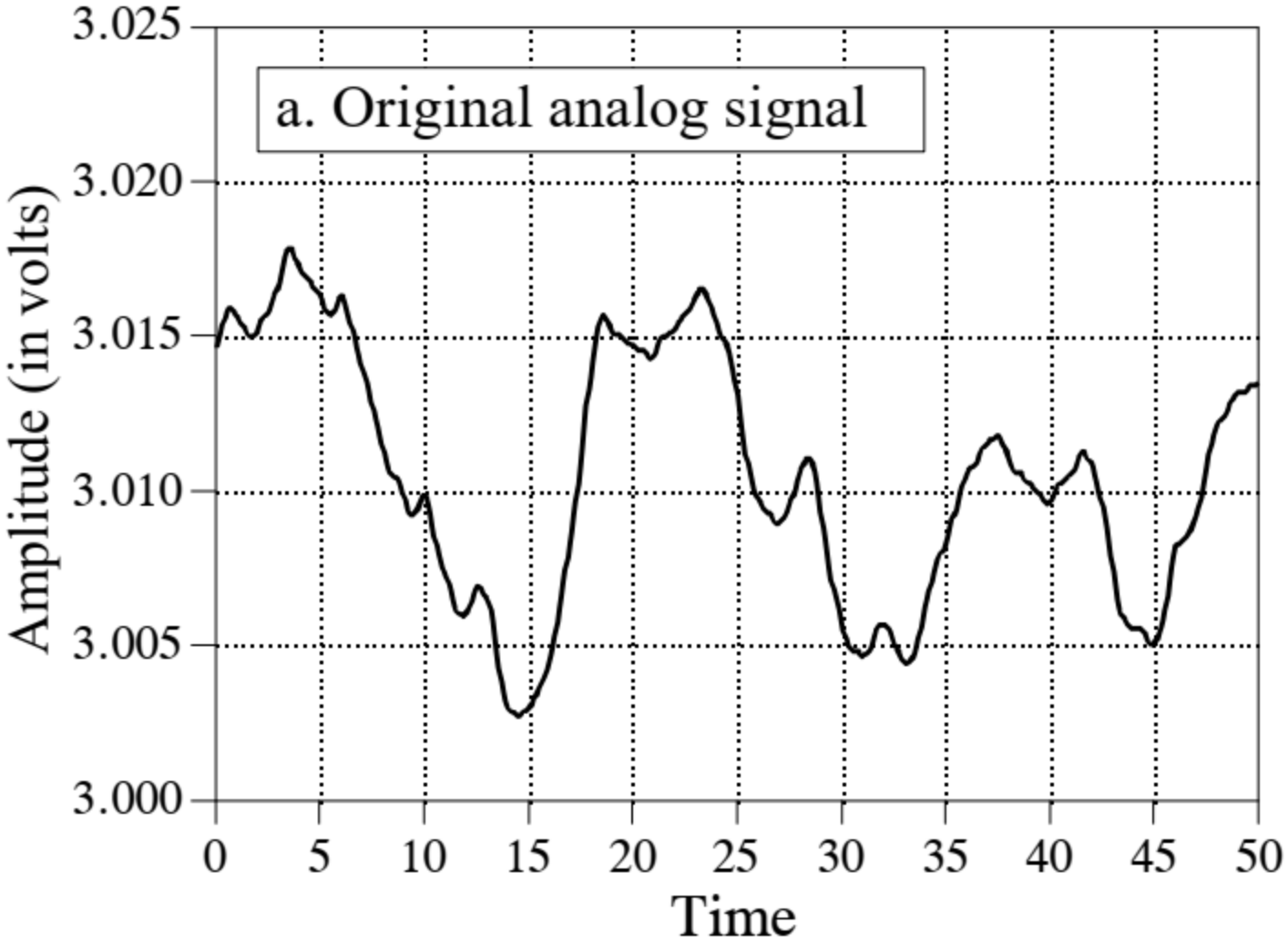
`norm(v)`	`sqrt(v @ v)` or `np.linalg.norm(v)`
`a & b`	`logical_and(a,b)`
`a | b`	`logical_or(a,b)`
`bitand(a,b)`	`a & b`
`bitor(a,b)`	`a | b`
`inv(a)`	`linalg.inv(a)`
`pinv(a)`	`linalg.pinv(a)`
`rank(a)`	`linalg.matrix_rank(a)`
`a\b`	`linalg.solve(a,b)` if `a` is square; `linalg.lstsq(a,b)` otherwise
`b/a`	Solve a.T x.T = b.T instead
`[U,S,V]=svd(a)`	`U, S, Vh = linalg.svd(a), V = Vh.T`
`chol(a)`	`linalg.cholesky(a).T`



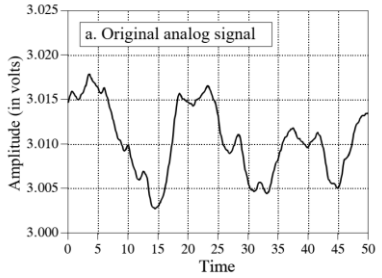
MATLAB

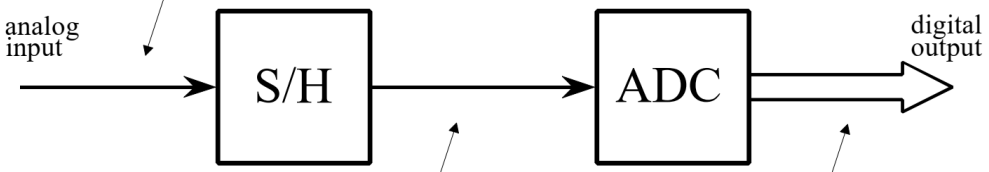
`[V,D]=eig(a)`	`D,V = linalg.eig(a)`
`[V,D]=eig(a,b)`	`V,D = np.linalg.eig(a,b)`
`[V,D]=eigs(a,k)`	
`[Q,R,P]=qr(a,0)`	`Q,R = scipy.linalg.qr(a)`
`[L,U,P]=lu(a)`	`L,U = scipy.linalg.lu(a)` or `LU,P=scipy.linalg.lu_factor(a)`
`conjgrad`	`scipy.sparse.linalg.cg`
`fft(a)`	`fft(a)`
`ifft(a)`	`ifft(a)`
`sort(a)`	`sort(a)` or `a.sort()`
`[b,I] = sortrows(a,i)`	`I = argsort(a[:,i]), b=a[I,:]`
`regress(y,X)`	`linalg.lstsq(X,y)`
`decimate(x, q)`	`scipy.signal.resample(x, len(x)/q)`
`unique(a)`	`unique(a)`
`squeeze(a)`	`a.squeeze()`

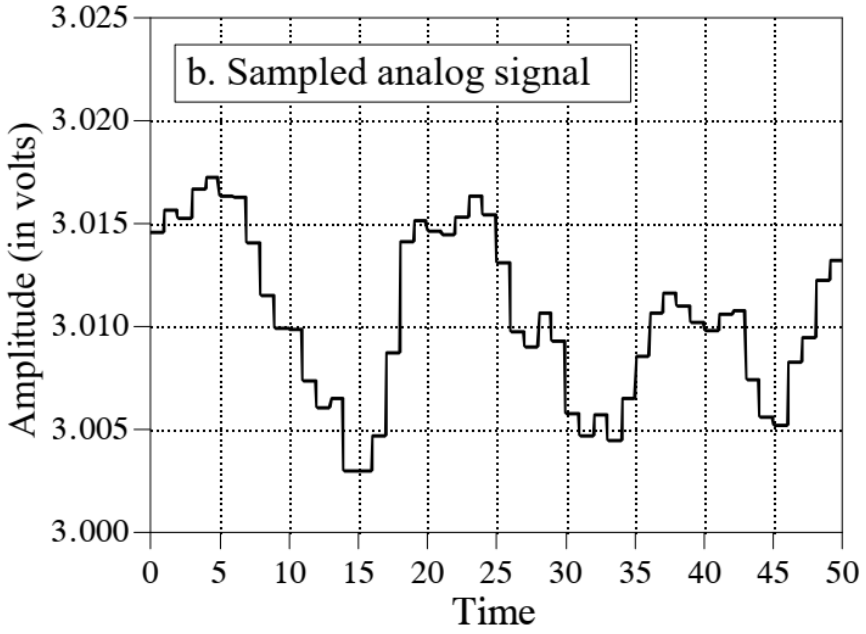
ADC

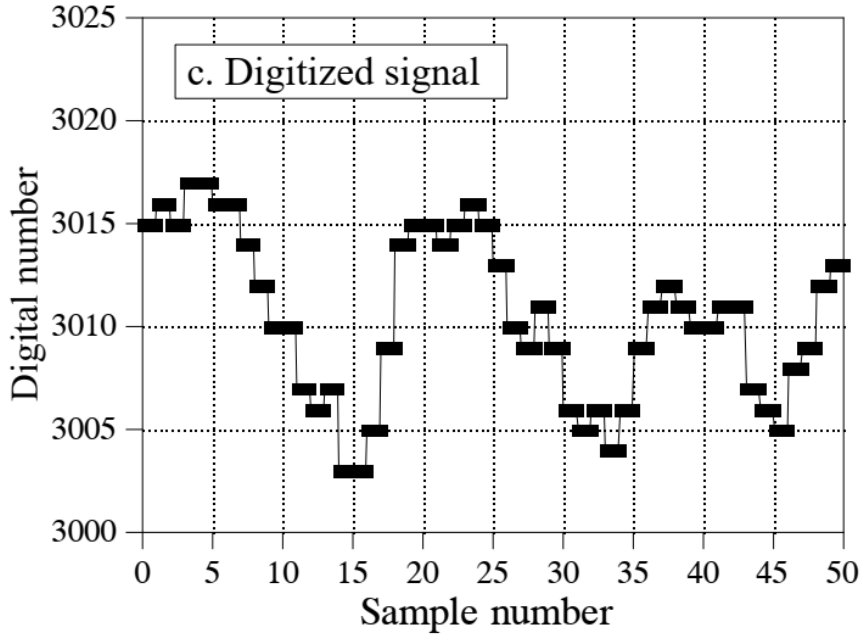

a. Original analog signal
Amplitude (in volts)
3.025
3.020
3.015
3.010
3.005
3.000
0
5
10
15
20
25
30
35
40
45
50
Time

ADC

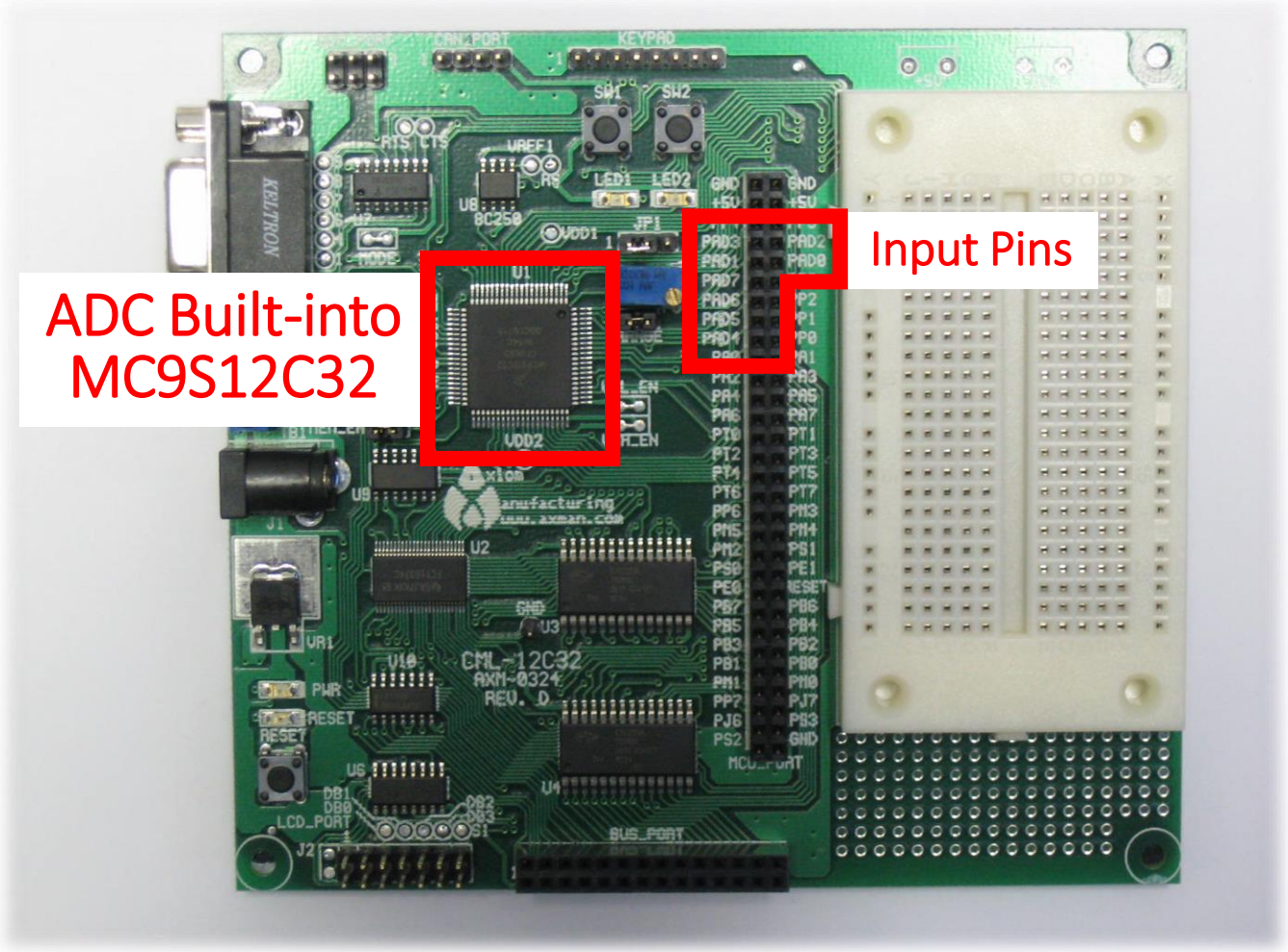

a. Original analog signal
Amplitude (in volts)
3.025
3.020
3.015
3.010
3.005
3.000
0 5 10 15 20 25 30 35 40 45 50
Time


analog input
S/H
ADC
digital output


b. Sampled analog signal
Amplitude (in volts)
3.025
3.020
3.015
3.010
3.005
3.000
0 5 10 15 20 25 30 35 40 45 50
Time


c. Digitized signal
Digital number
3025
3020
3015
3010
3005
3000
0 5 10 15 20 25 30 35 40 45 50
Sample number

ADC

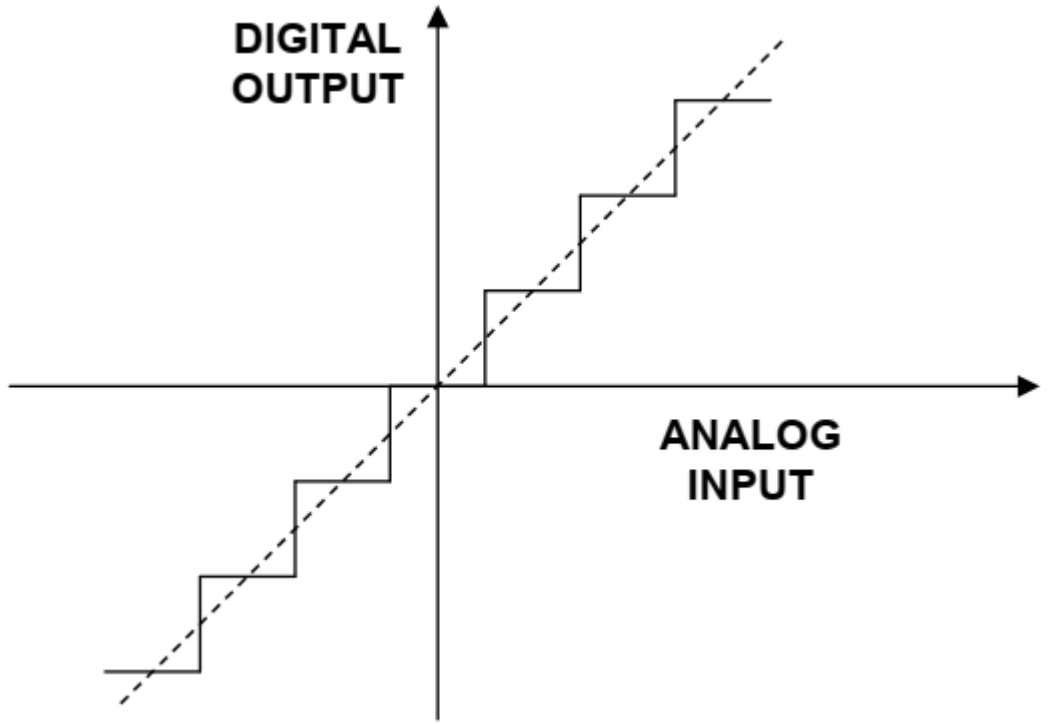


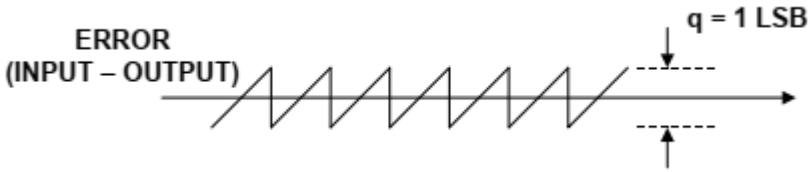
ADC

ADC Sampling Rate (500MHz)

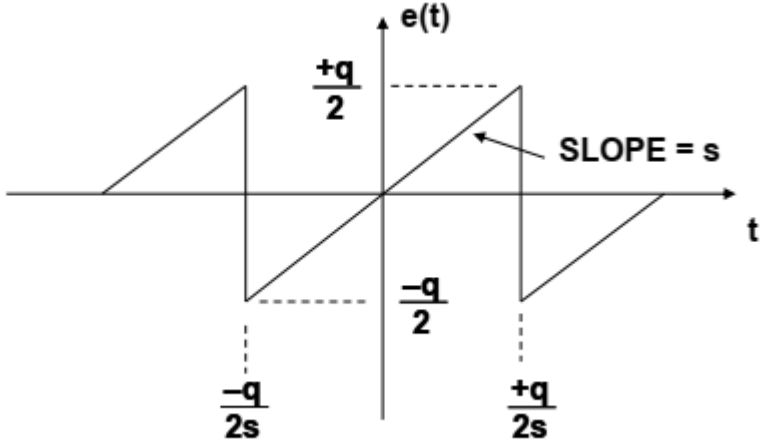
ADC Quantization Bits (12 bit)

ADC Quantization Noise





$$e(t) = st,\ -q/2s < t < +q/2s.$$



ADC Quantization Noise

$$\overline{e^2(t)} = \frac{s}{q}\int_{-q/2s}^{+q/2s}(st)^2\,dt$$

$$\overline{e^2(t)} = \frac{q^2}{12}$$

$$\text{rms quantization noise} = \sqrt{\overline{e^2(t)}} = \frac{q}{\sqrt{12}}$$

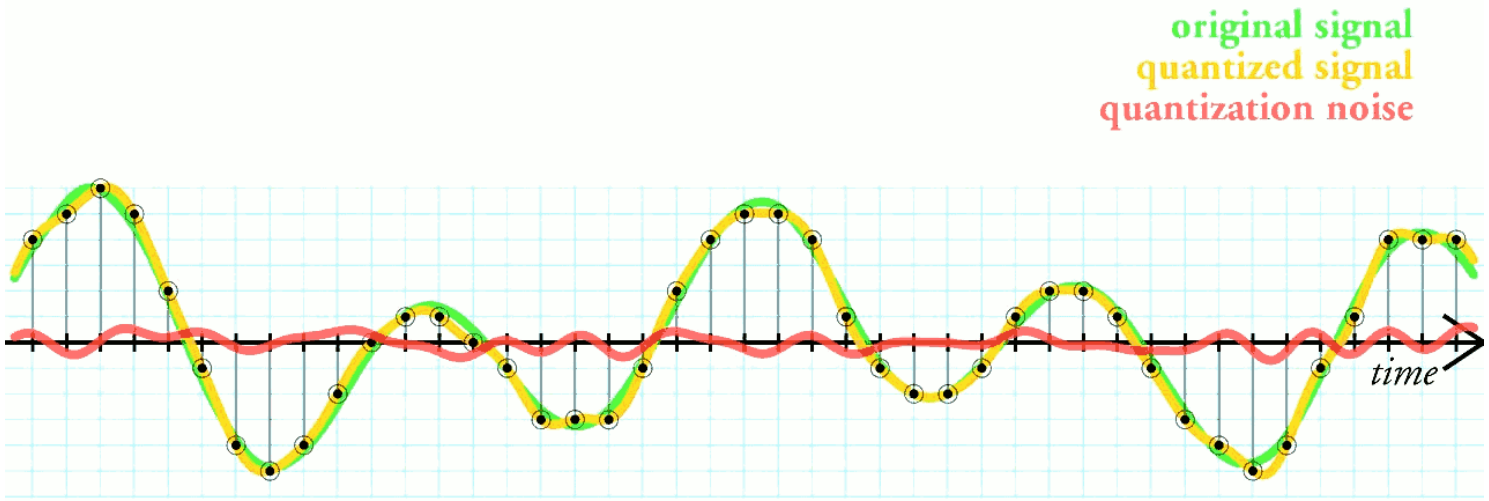
ADC Quantization Noise

$$\overline{e^2(t)} = \frac{s}{q}\int_{-q/2s}^{+q/2s}(st)^2\,dt$$

$$\overline{e^2(t)} = \frac{q^2}{12}$$

$$\text{rms quantization noise} = \sqrt{\overline{e^2(t)}} = \frac{q}{\sqrt{12}}$$

ADC Quantization SNR (N Bit)



full-scale input sinewave: $$v(t) = \frac{q2^N}{2}\sin(2\pi ft).$$

$$\text{SNR} = 20\log_{10}\frac{\text{rms value of FS input}}{\text{rms value of quantization noise}}$$

ADC Quantization SNR (N Bit)

full-scale input sinewave: $v(t) = \frac{q2^N}{2}\sin(2\pi ft).$

$$\text{SNR} = 20\log_{10}\frac{\text{rms value of FS input}}{\text{rms value of quantization noise}}$$

$$\text{SNR} = 20\log_{10}\left[\frac{q2^N/2\sqrt{2}}{q/\sqrt{12}}\right] = 20\log_{10}2^N + 20\log_{10}\sqrt{\frac{3}{2}}$$

$$\text{SNR} = 6.02N + 1.76\text{dB}, \qquad \text{over the dc to } f_s/2 \text{ bandwidth.}$$

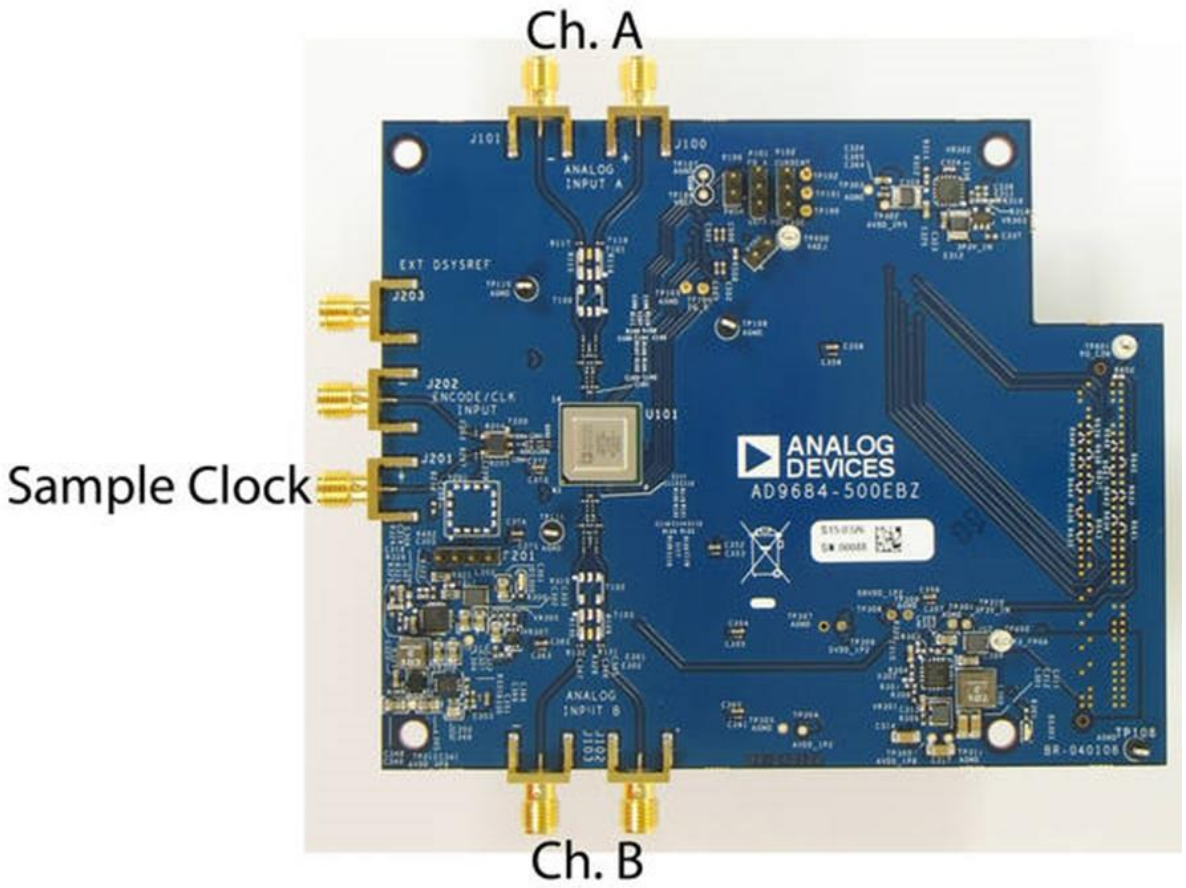
ADC Quantization SNR (N Bit)

$$SNR = 6.02N + 1.76dB$$

- N = 2 SNR = 13.8 dB
- N = 3 SNR = 19.8 dB
- N = 6 SNR = 37.9 dB
- N = 8 SNR = 49.9 dB
- N = 10 SNR = 61.96 dB
- N = 12 SNR = 74 dB
- N = 14 SNR = 86 dB
- N = 16 SNR = 98 dB

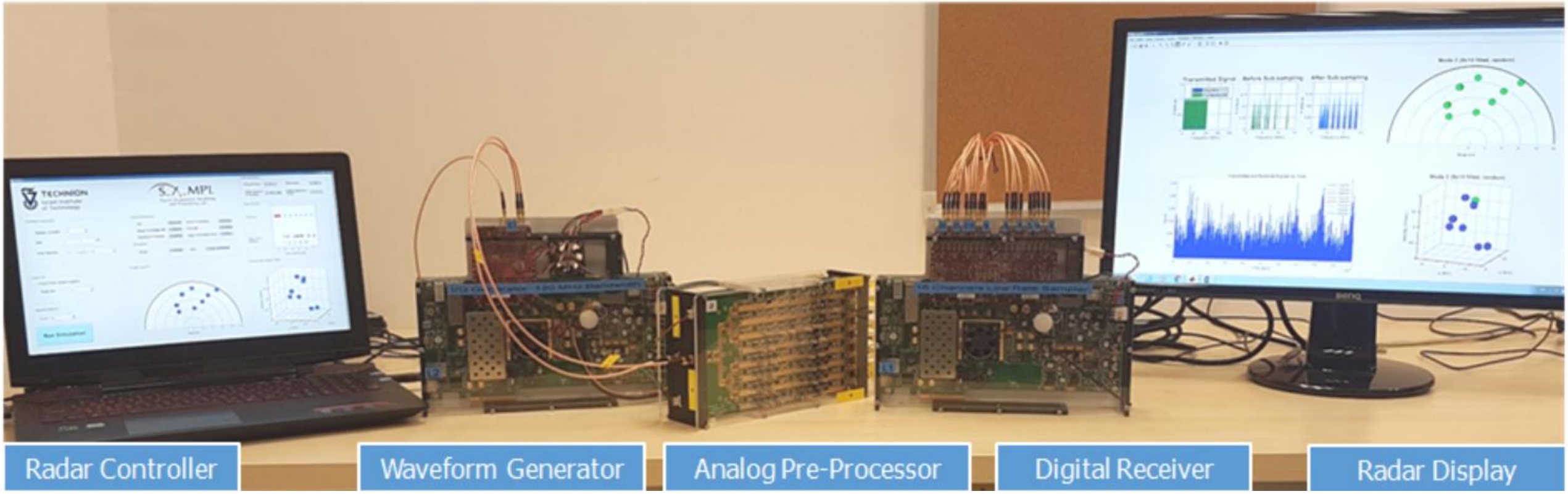
ADC

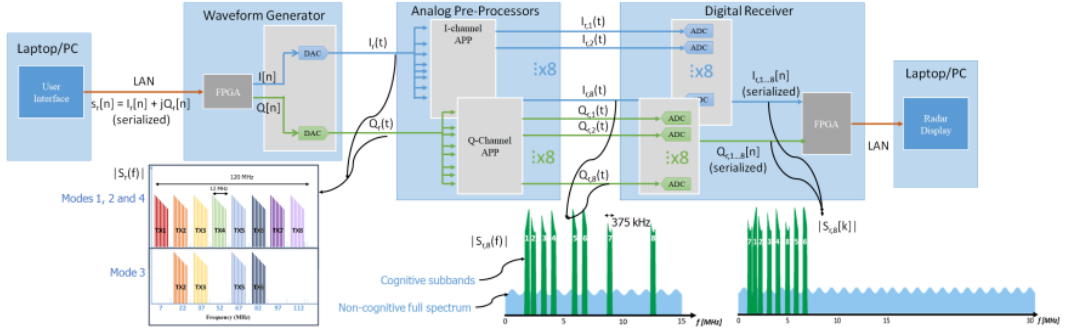
AD9684



Model	Description	Price
AD9684-500EBZ Production	Evaluation Board	**$795.00**

ADC in SUMMeR (Sub-Nyquist colocated MiMo Radar)





ADC

AD9684

ANALOG DEVICES

14-Bit, 500 MSPS LVDS, Dual Analog-to-Digital Converter

APPLICATIONS

Communications
Diversity multiband, multimode digital receivers
3G/4G, TD-SCDMA, W-CDMA, MC-GSM, LTE
General-purpose software radios
Ultrawideband satellite receiver
Instrumentation (spectrum analyzers, network analyzers, integrated RF test solutions)
Radar
Digital oscilloscopes
High speed data acquisition systems
DOCSIS CMTS upstream receiver paths
HFC digital reverse path receivers

FEATURES

Parallel LVDS (DDR) outputs
1.1 W total power per channel at 500 MSPS
SFDR = 85 dBFS at 170 MHz f_{IN} (500 MSPS)
SNR = 68.6 dBFS at 170 MHz f_{IN} (500 MSPS)
ENOB = 10.9 bits at 170 MHz f_{IN}
DNL = ±0.5 LSB
INL = ±2.5 LSB
Noise density = −153 dBFS/Hz at 500 MSPS
1.25 V, 2.50 V, and 3.3 V supply operation

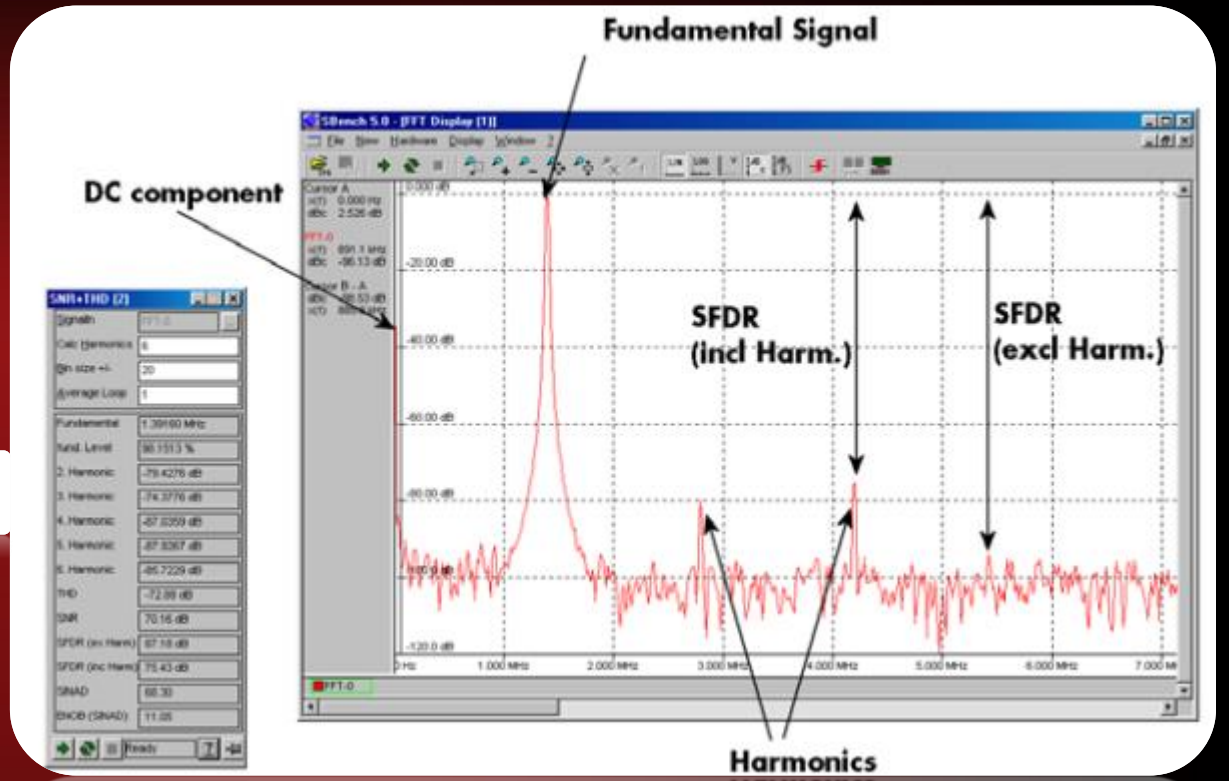
ADC

$$SNR = 20 * \log([Fundamental] / SQRT(SUM(SQR([Noise]))))$$

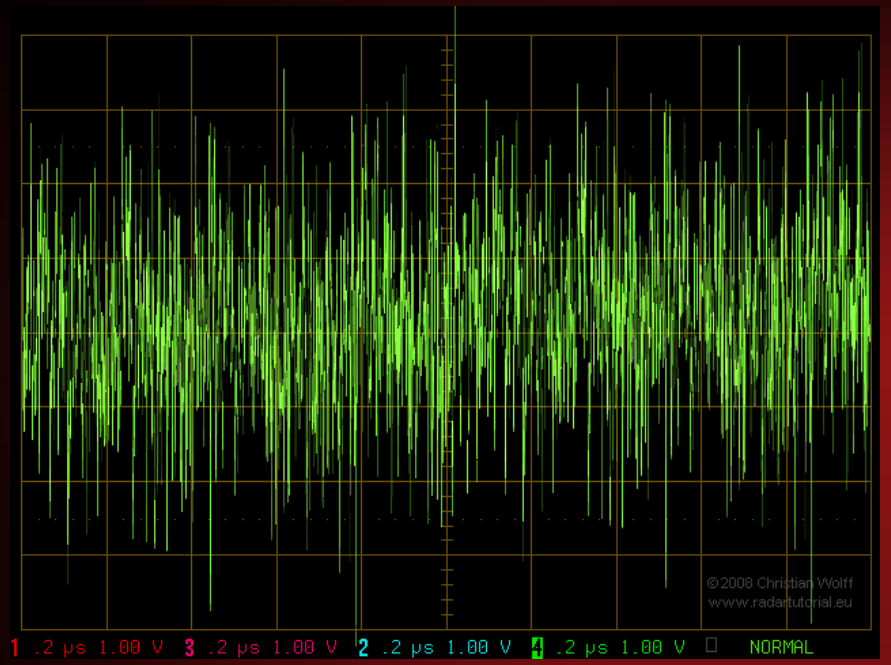
$$THD = 20 * \log(SQRT(SUM(SQR([Harmonics]))) / [Fundamental])$$

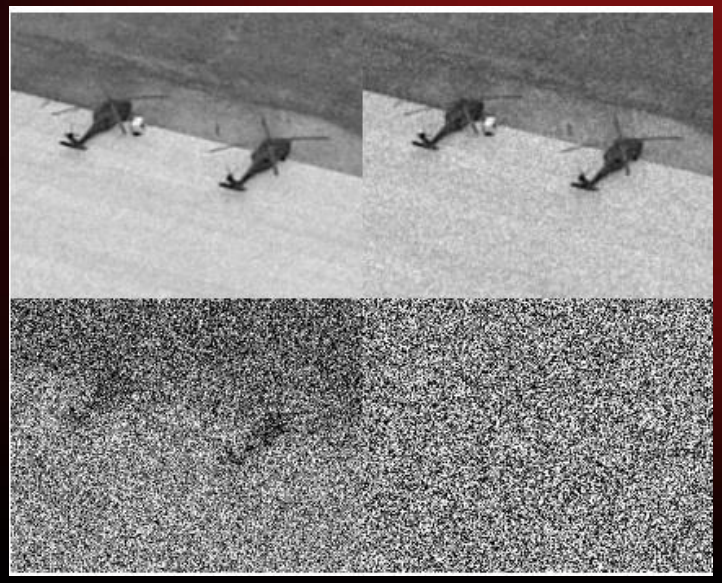
$$SINAD = 20 * \log([Fundamental] / SQRT(SUM(SQR([Noise + Harmonics]))))$$

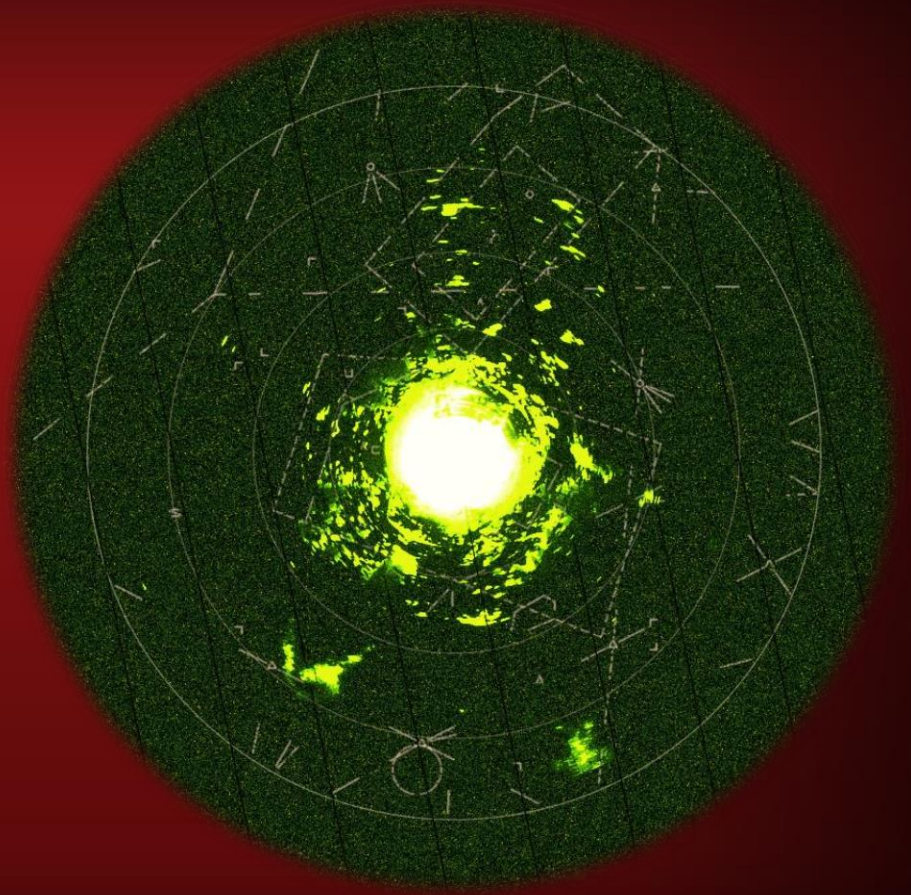
$$ENOB = (SINAD - 1.76) / 6.02$$



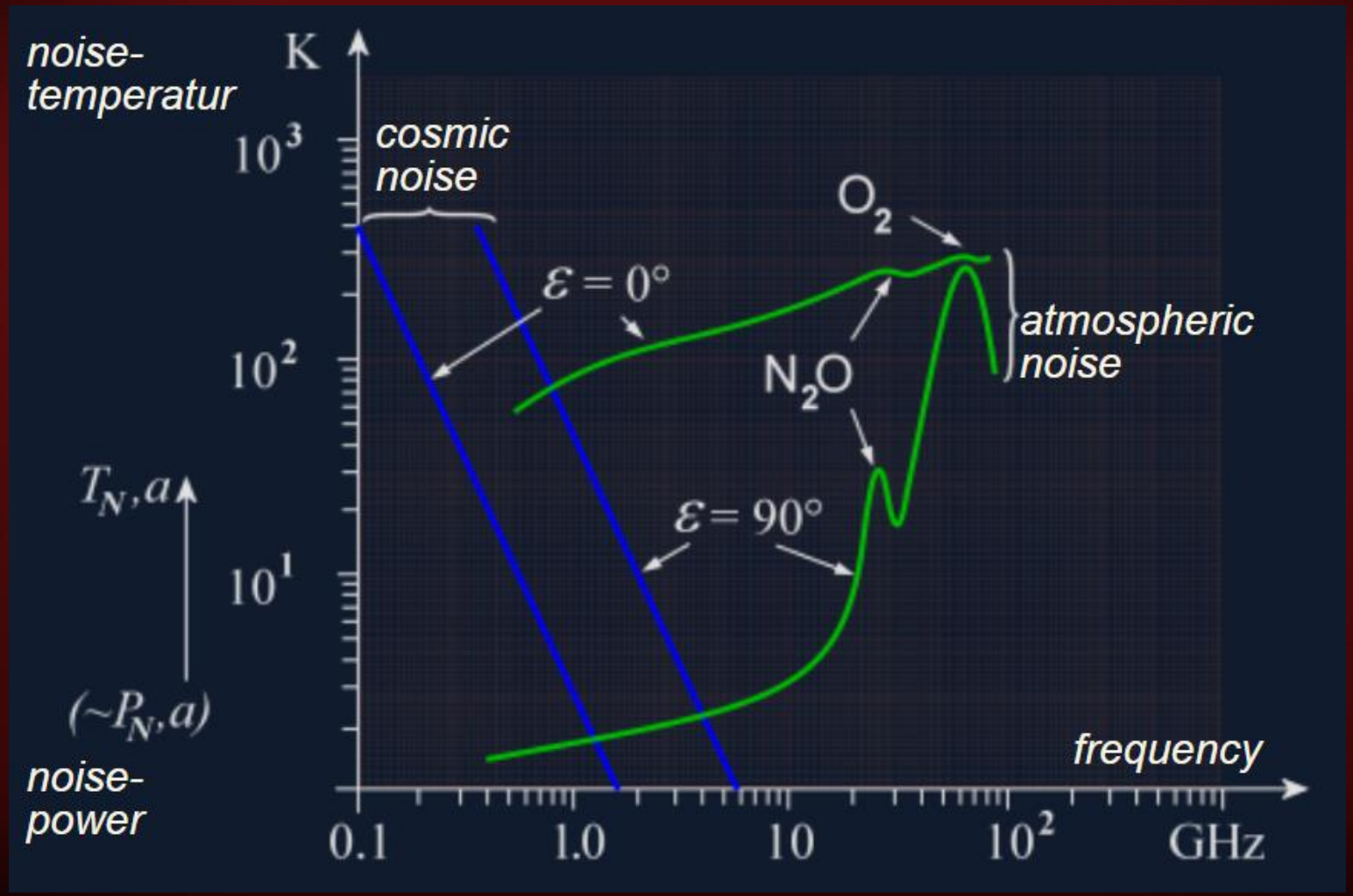
Noise


© 2008 Christian Wolff
www.radartutorial.eu
1 .2 µs 1.00 V 3 .2 µs 1.00 V 2 .2 µs 1.00 V 4 .2 µs 1.00 V NORMAL

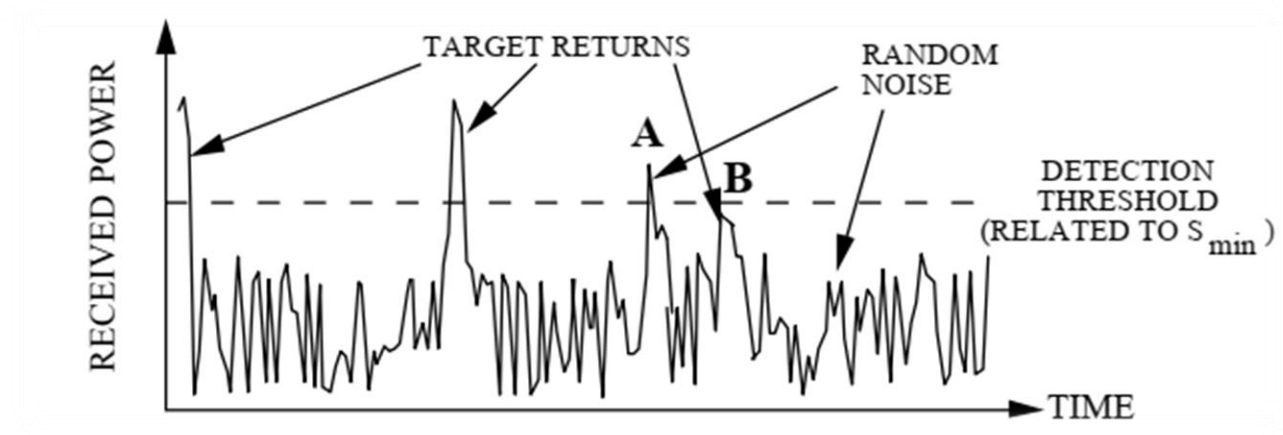




Noise


noise-
temperatur
K
10^3
10^2
10^1
cosmic
noise
O_2
$\varepsilon = 0°$
N_2O
atmospheric
noise
T_N, a
$\varepsilon = 90°$
$(\sim P_N, a)$
noise-
power
frequency
0.1
1.0
10
10^2
GHz

Thermal Noise



Consider a receiver at the standard temperature, T_o degrees Kelvin (K). Over a range of frequencies of bandwidth B_n (Hz) the available noise power is

$$N_o = kT_oB_n$$

where $k_B = 1.38\times10^{-23}$ (Joules/K) is Boltzman's constant.

$$\text{SNR} = \frac{P_r}{N_o} = \frac{P_tG_tG_r\sigma\lambda^2G_pL}{(4\pi)^3R^4k_BT_sB_n}$$

Thermal Noise

$$N_o = kT_oB_n$$

```
K=1.38*1e-23
```

```
B = 1000 #Hz
```

```
T = 290
```

```
import math
N0 = K*T*B
N0dB = 10*math.log10(N0)
```

```
print("Noise Power for BW: 1 KHz = {0} dB = {1} dBm".format(N0dB,N0dB+30))
```

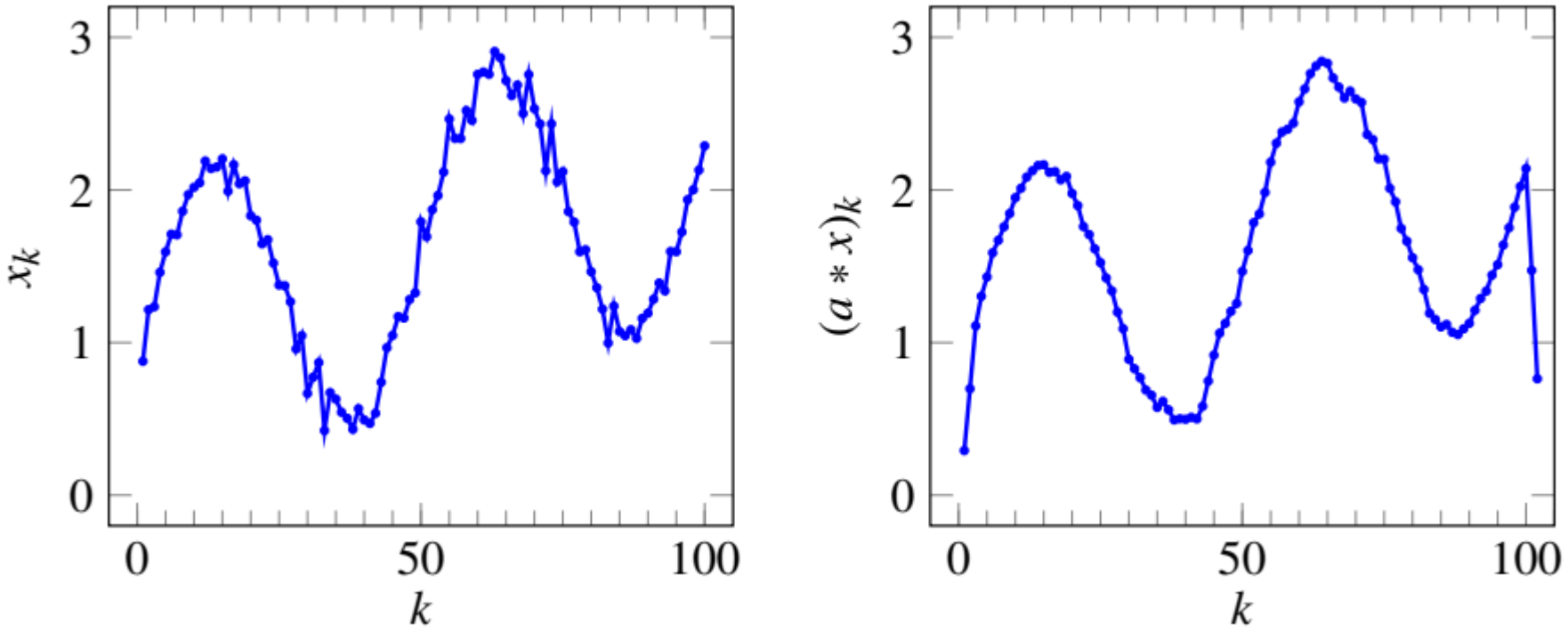
```
Noise Power for BW: 1 KHz = -173.97722915699808 dB = -143.97722915699808 dBm
```

System Noise

$$\text{Noise floor}_{\text{dBm}} = 10\log_{10}(k \times T_0 \times 1000) + \text{NF} + 10\log_{10}\text{BW}$$

Moving Average Filter

$$y_k = \frac{1}{3}(x_k + x_{k-1} + x_{k-2}), \quad k = 1, 2, \ldots, n+2$$



Previous exposure

- linear system theory for continuous-time signals and systems including Fourier and Laplace Transforms

Class Review

General System

$x(n) \longrightarrow y(n)$

Special Class:

Linear $\dot{\varepsilon}$

Shift-invariant

LSI

Linearity

If $x_1(n) \longrightarrow y_1(n)$

$\dot{\varepsilon}$ $x_2(n) \longrightarrow y_2(n)$

then: $ax_1(n) + bx_2(n)$

$\longrightarrow ay_1(n) + by_2(n)$

$\sum a_k x_k(n) \longrightarrow$

$\sum a_k y_k(n)$

Shift-invariance

$x(n-n_0) \longrightarrow y(n-n_0)$

$\delta(n) \longrightarrow h(n)$

(unit sample response)

$\delta(n-k) \longrightarrow h(n-k)$

MIT DSP Course

Class Review

$$x(n)=\sum_{k=-\infty}^{+\infty} x(k)\,\underbrace{\delta(n-k)}$$

$$y(n)=\sum_{k=-\infty}^{+\infty} x(k)\,h(n-k)$$

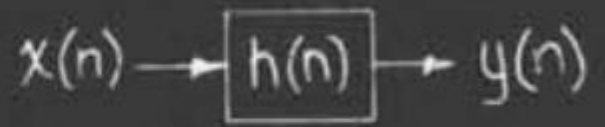
Convolution sum

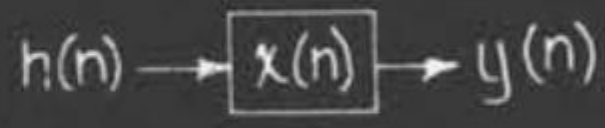
$n-k=r\;;\quad k=n-r$

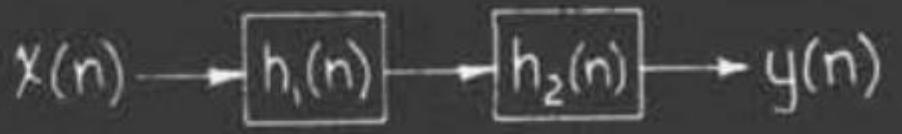
$$y(n)=\sum_{r} x(n-r)\,h(r)$$

$$y(n)=x(n)*h(n)$$

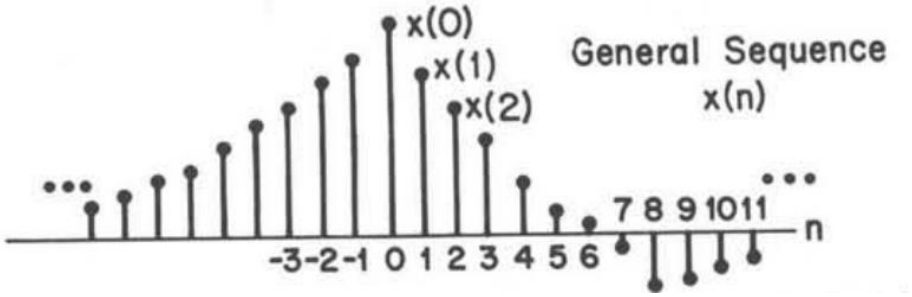
$$=h(n)*x(n)$$

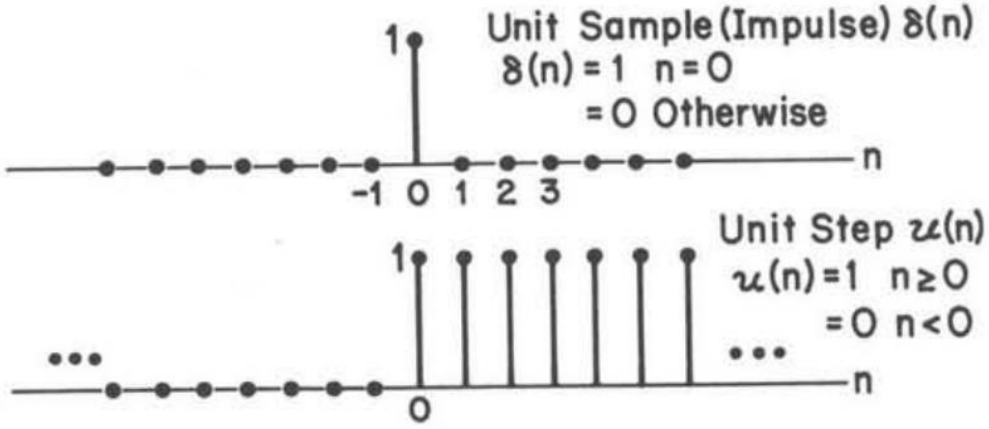




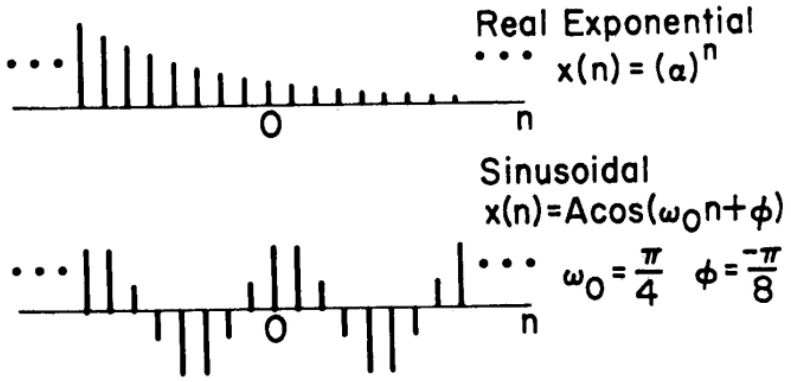


Class Review

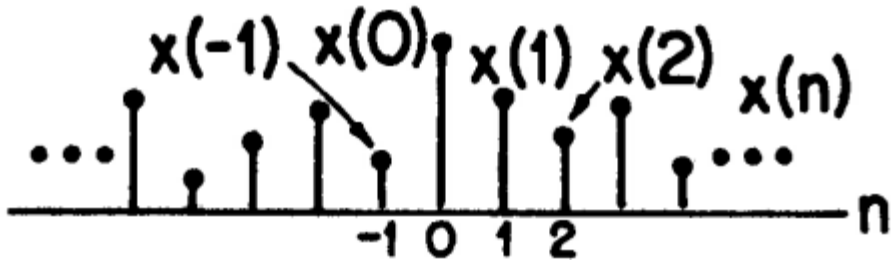




$$\delta(n) = u(n) - u(n-1)$$

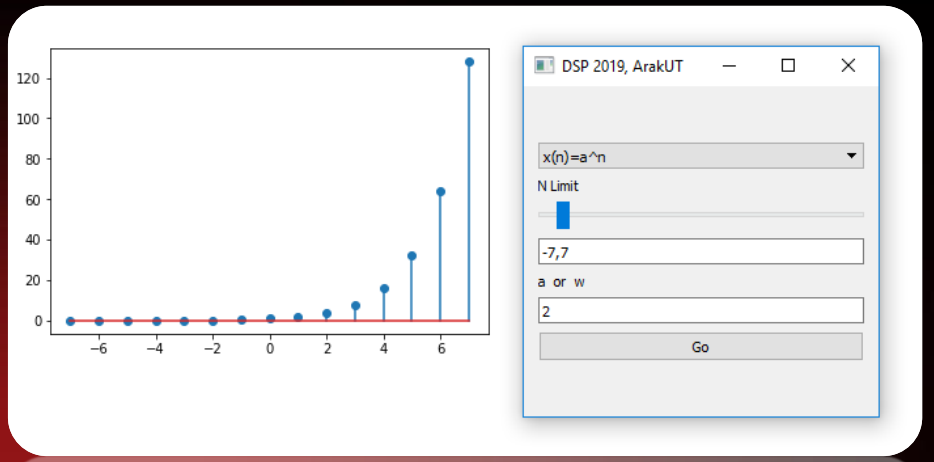


Class Review



$$x(n) = x(0)\delta(n) + x(1)\delta(n-1) + x(-1)\delta(n+1) + \cdots = \sum_{k=-\infty}^{+\infty} x(k)\delta(n-k)$$

DSP+Python UI




```
from PyQt5.QtWidgets import
QApplication,QWidget,QPushButton,QLineEdit,QSlider,QComboBox,QVBoxLayout,QHBoxLayout,QLabel
from PyQt5.QtCore import *
import matplotlib.pyplot as plt
import math
def sliderslot(value):
    dspLineEdit.setText(str(-(value+1))+","+str(value+1))
def pbslot():
    case = dspCombo.currentIndex()
    Limits = dspLineEdit.text().split(",")
    if len(Limits)!=2:
        return
    N1 = int(Limits[0])
    N2 = int(Limits[1])
    a  = float(dspLineEdita.text())
    nv = range(N1,N2+1)
    if case==0:
        plt.stem(nv,[a**n for n in nv])
    elif case==1:
        plt.stem(nv,[math.cos(a*n) for n in nv])
    elif case==2:
        nv = range(0,300)
        plt.stem(nv,[math.cos(.01*n+.001*n*n) for n in nv])
        plt.show()
        plt.plot(nv,[math.cos(.01*n+.001*n*n) for n in nv])
    plt.show()

app = QApplication([])
frame = QWidget()
frame.setGeometry(100,100,800,400)
frame.setWindowTitle("DSP 2019, ArakUT")
pb = QPushButton("Go",frame)
dspSlider = QSlider(Qt.Horizontal,frame)
dspSlider.valueChanged.connect(sliderslot)
```

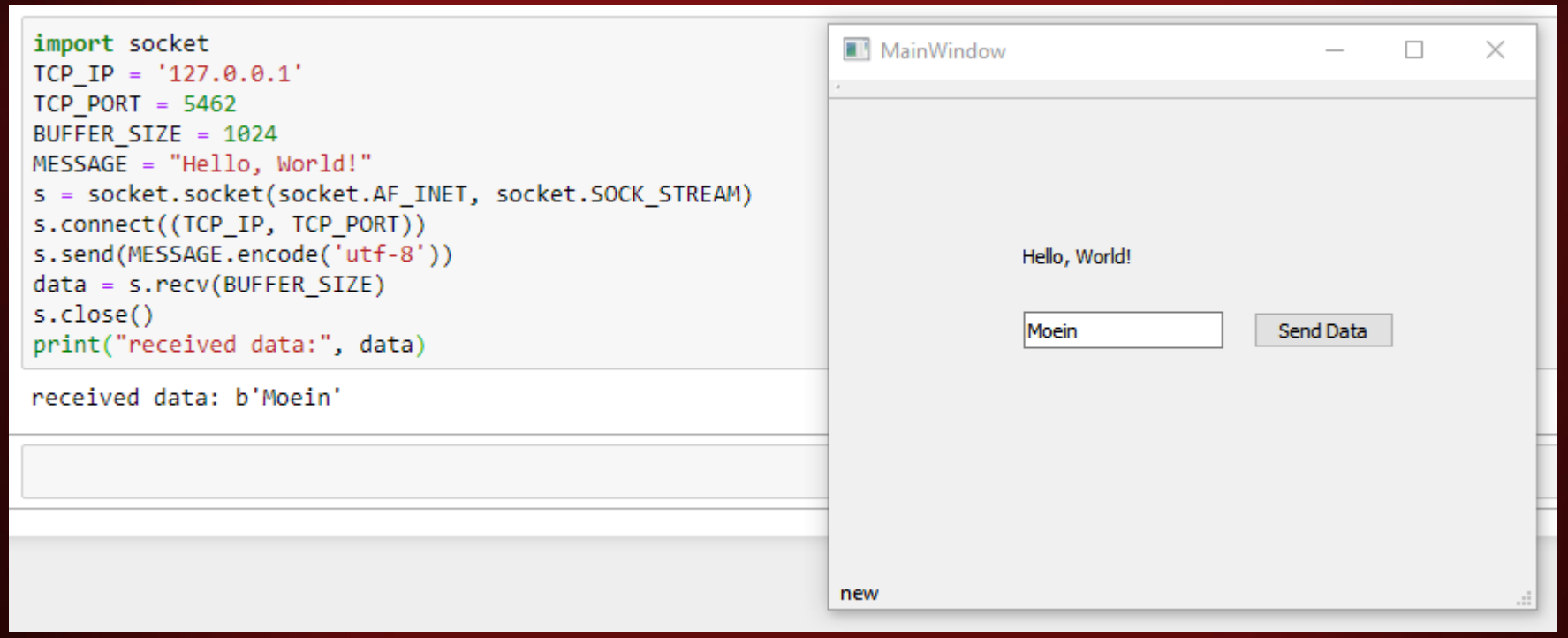
```
pb.clicked.connect(pbslot)
dspLineEdit=QLineEdit("-1,1",frame)
dspLineEdita=QLineEdit("0.5",frame)
dspCombo = QComboBox(frame)
dspCombo.addItem("x(n)=a^n")
dspCombo.addItem("x(n)=cos(w*n)")
dspCombo.addItem("LFM Radar waveform x(n)=cos(w1*n+B*n^2)")
mainLayout = QVBoxLayout(frame)
mainLayout.addStretch(1)
mainLayout.addWidget(dspCombo)
mainLayout.addWidget(QLabel("N Limit"))
mainLayout.addWidget(dspSlider)
mainLayout.addWidget(dspLineEdit)
mainLayout.addWidget(QLabel("a  or  w"))
mainLayout.addWidget(dspLineEdita)
mainLayout.addWidget(pb)
mainLayout.addStretch(1)
frame.show()
app.exec_()
```

TCP Connection in Python and C++

```
{
    ui->setupUi(this);
    server = new QTcpServer(this);
    connect(server, SIGNAL(newConnection()),this, SLOT(newConnection()));
    server->listen(QHostAddress("127.0.0.1"),5462);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::readyRead(){
    ui->label->setText(socket->readAll());
}
void MainWindow::newConnection()
{
    ui->statusBar->showMessage("new");
    socket = server->nextPendingConnection();
    socket->setReadBufferSize(1024);
    connect(socket, SIGNAL(readyRead()),this, SLOT(readyRead()));
}

void MainWindow::on_pushButton_clicked()
{
    socket->write(ui->lineEdit->text().toLatin1());
```

```
public:
    explicit MainWindow(QWidget *parent = nul
    ~MainWindow();
    QTcpServer *server;
    QTcpSocket *socket;
private:
    Ui::MainWindow *ui;
public slots:
    void newConnection();
    void readyRead();
private slots:
    void on_pushButton_clicked();
```

TCP Connection in Python and C++



Class Review

$x(n) \rightarrow T[\,] \rightarrow y(n)$

general

$y(n) = T[x(n)]$

LSI

$$y(n) = \sum_{k=-\infty}^{+\infty} x(k)\,h(n-k)$$

$$= \sum_{k=-\infty}^{+\infty} h(k)\,x(n-k)$$

Convolution Sum

Stability

general: If $x(n)$ bounded

ie $|x(n)| < \infty$ all n

then $y(n)$ bounded

ie $|y(n)| < \infty$ all n

LSI

$$\sum_{k=-\infty}^{+\infty} |h(k)| < \infty$$

$h(n) = 2^n\,u(n)$ — unstable

$h(n) = \left(\frac{1}{2}\right)^n u(n)$ — stable

Causality

$y(n)$ for $n = n_1$ depends on $x(n)$ only for $n \le n_1$

LSI

$h(n) = 0 \quad n < 0$

$h(n) = (2)^n\,u(-n)$

noncausal

stable

Class Review

$x(n) \rightarrow T[\,] \rightarrow y(n)$

general

$y(n) = T[x(n)]$

LSI

$$y(n) = \sum_{k=-\infty}^{+\infty} x(k)\,h(n-k)$$

$$= \sum_{k=-\infty}^{+\infty} h(k)\,x(n-k)$$

Convolution Sum

Stability

general: If $x(n)$ bounded

ie $|x(n)| < \infty$ all n

then $y(n)$ bounded

ie $|y(n)| < \infty$ all n

LSI

$$\sum_{k=-\infty}^{+\infty} |h(k)| < \infty$$

$h(n) = 2^n\,u(n)$ — unstable

$h(n) = \left(\frac{1}{2}\right)^n u(n)$ — stable

Causality

$y(n)$ for $n = n_1$ depends on $x(n)$ only for $n \le n_1$

LSI

$h(n) = 0 \quad n < 0$

$h(n) = (2)^n\,u(-n)$

noncausal

stable

Class Review

Linear Constant Coefficient Difference Equation

N^{th} order

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{r=0}^{M} b_r x(n-r)$$

$N=0 \quad a_0=1$

$$y(n) = \sum_{r=0}^{M} b_r x(n-r)$$

$$h(n) = b_n \quad n=0,1,\dots,M$$
$$= 0 \text{ otherwise}$$

$N \neq 0 \quad a_0 = 1$

$$y(n) = \sum_{r=0}^{M} b_r x(n-r) - \sum_{k=1}^{N} a_k y(n-k)$$

First-order

$$y(n) + a y(n-1) = x(n)$$
$$x(n) = \delta(n)$$

assume $y(n)=0 \quad n<0$

$$y(n) = \delta(n) + a y(n-1)$$

$y(-1)=0$, $y(0)=1$, $y(1)=a$, $y(2)=a^2$ → $a^n u(n)$, $|a|<1$, Stable

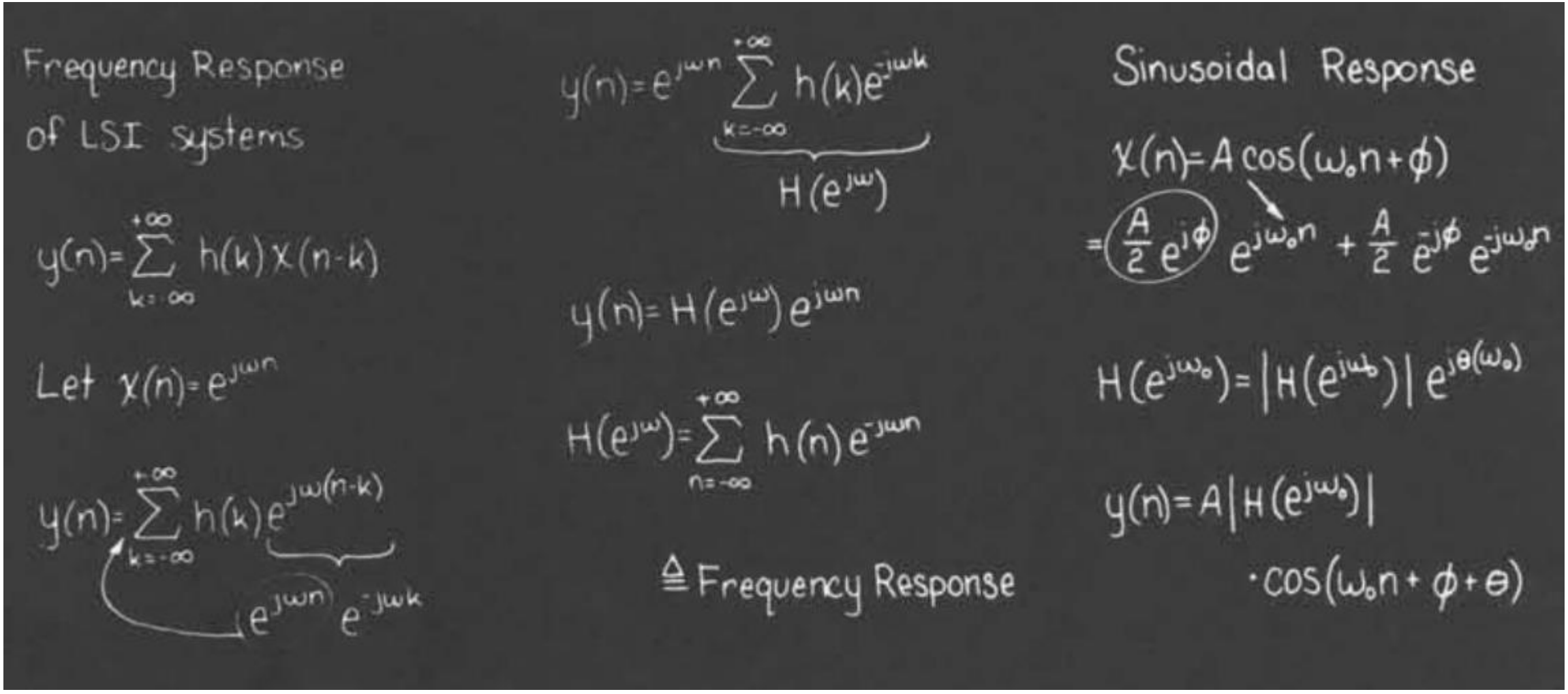
$x(n)=\delta(n)$

assume $y(n)=0 \quad n>0$

$$y(n-1) = a^{-1}\left[y(n) - \delta(n)\right]$$

$y(1)=0$, $y(0)=0$, $y(-1)=-a^{-1}$, $y(-2)=-a^{-2}$ → $-a^n u(-n-1)$, $|a|<1$, unstable

Class Review



Class Review

Example

$y(n) - a y(n-1) = x(n)$

causal

$h(n) = a^n u(n) \quad 0 < a < 1$

$$H(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} a^n u(n) e^{-j\omega n}$$

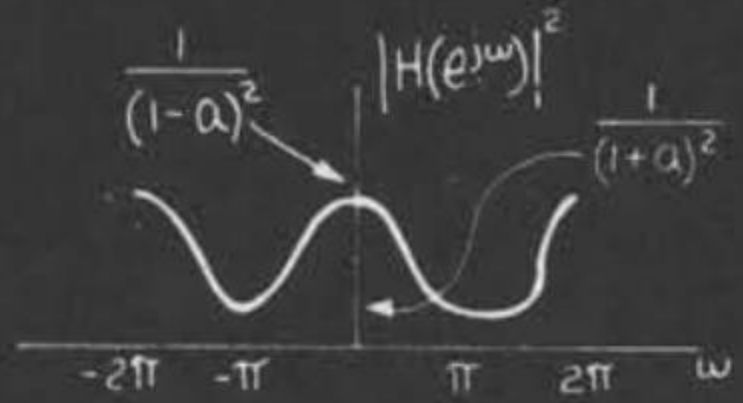
$$= \sum_{n=0}^{+\infty} (a e^{-j\omega})^n \qquad \sum_{n=0}^{\infty} \alpha^n$$

$$= \frac{1}{1 - a e^{-j\omega}}$$

$$|H(e^{j\omega})|^2 = \frac{1}{1 - a e^{j\omega}} \cdot \frac{1}{1 - a e^{-j\omega}}$$

$$= \frac{1}{1 + a^2 - 2a\cos\omega}$$

$$\angle H(e^{j\omega}) = \tan^{-1}\left[\frac{-a\sin\omega}{1 - a\cos\omega}\right]$$



Properties of Freq Resp

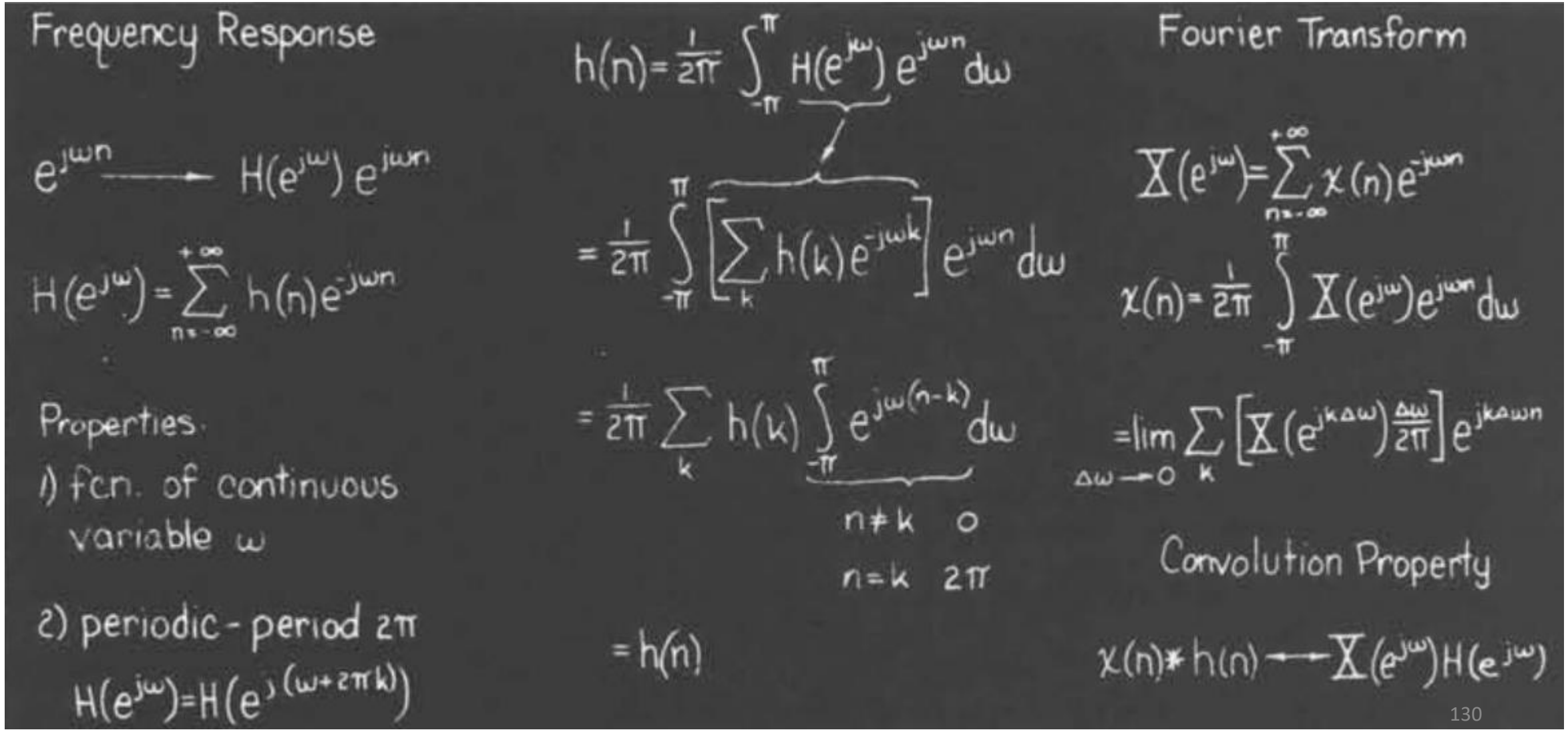
1. Function of continuous variable ω
2. Periodic function of ω period 2π

$$e^{j(\omega + 2\pi k)n} = e^{j\omega n} \underbrace{e^{j2\pi k n}}_{1}$$

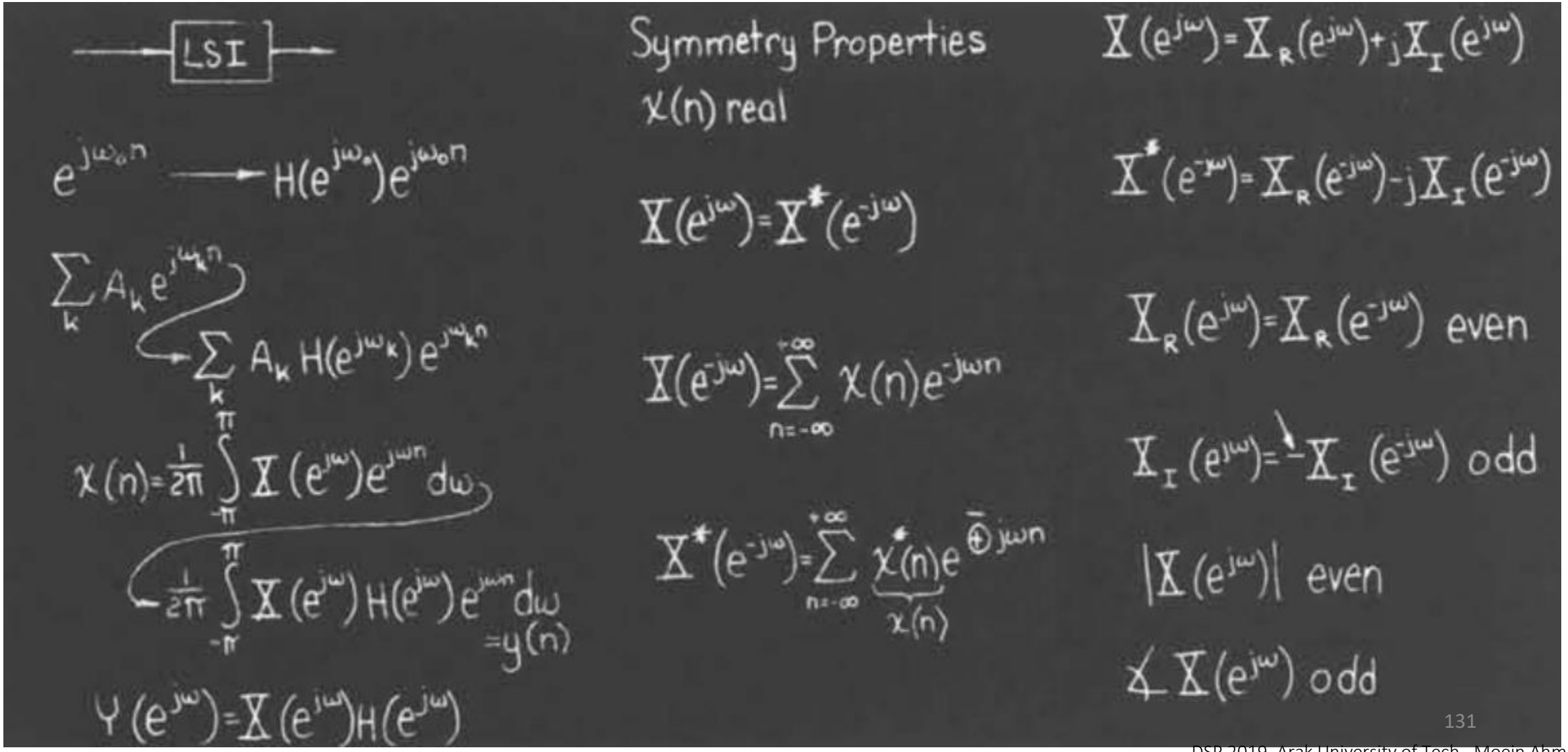
Generalization

The Fourier Transform

Class Review



Class Review



Class Review

$x_A(t) \rightarrow \boxed{\times I} \rightarrow \tilde{x}_A(t)$

$\tilde{X}_A(j\Omega) = \int_{-\infty}^{+\infty} \tilde{x}_A(t) e^{-j\Omega t} dt$

$\tilde{x}_A(t) \rightarrow \boxed{C/D} \rightarrow x(n) = x_A(nT)$

$\tilde{x}_A(t) = x_A(t) \cdot \sum_{n=-\infty}^{+\infty} u_0(t-nT)$

$= \sum_{n=-\infty}^{+\infty} x_A(nT) u_0(t-nT)$

$= \sum_{n=-\infty}^{+\infty} x_A(nT) \int_{-\infty}^{+\infty} e^{-j\Omega t} u_0(t-nT) dt$

$\tilde{X}_A(j\Omega) = \frac{1}{T} \sum_{r=-\infty}^{+\infty} X_A\left(j\Omega + \frac{2\pi r}{T}\right)$

$= \sum_{n=-\infty}^{+\infty} x_A(nT) e^{-jn\Omega T}$

$\tilde{X}_A(j\Omega) = X_A(j\Omega) * P(j\Omega)$

$= \sum_{n=-\infty}^{+\infty} x_A(nT) e^{-jn\Omega T}$

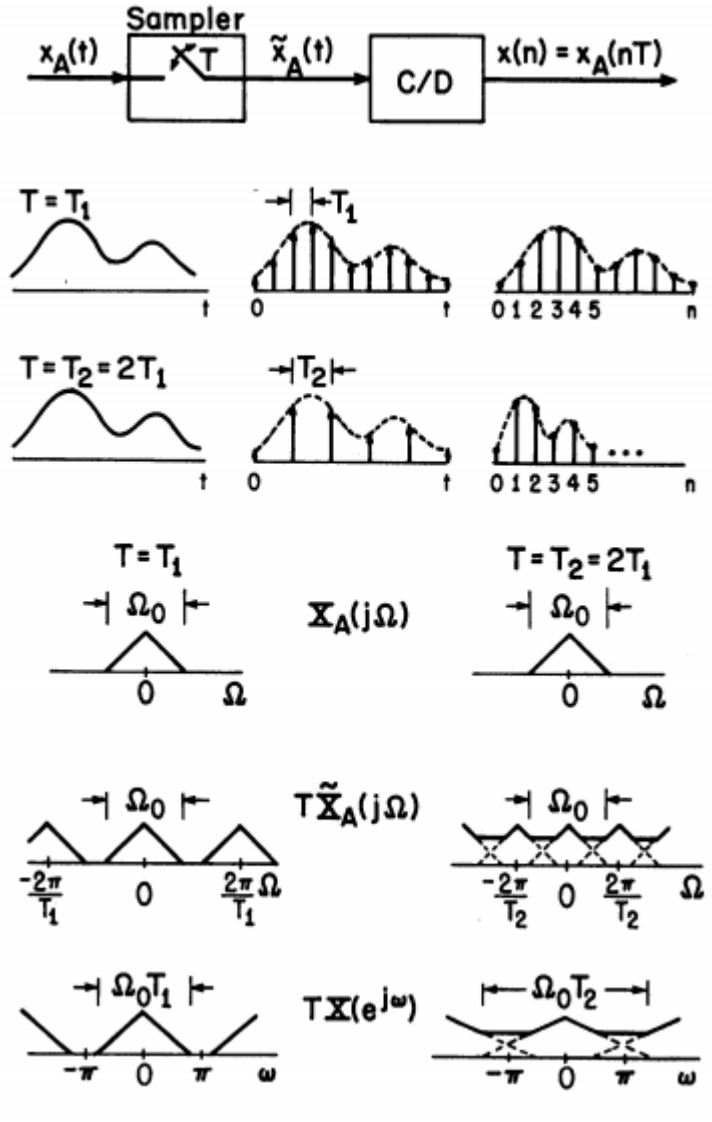
$X(e^{j\omega}) = \sum_{n} x_A(nT) e^{-j\omega n}$

$= \frac{1}{T} \sum_{r=-\infty}^{+\infty} X_A\left(j\Omega + j\frac{2\pi r}{T}\right)$

$X(e^{j\omega}) = \tilde{X}_A(j\Omega)\Big|_{\Omega T = \omega}$

$= \frac{1}{T} \sum_{n=-\infty}^{+\infty} X_A\left(j\Omega + j\frac{2\pi r}{T}\right)^{\Omega = \omega/T}$

Class Review



Class Review

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x(n) e^{-j\omega n}$$

$$|X(e^{j\omega})| = \left|\sum_{-\infty}^{+\infty} x(n) e^{-j\omega n}\right|$$

$$\leq \sum_{-\infty}^{+\infty} |x(n)|\,|e^{-j\omega n}|, \quad |e^{-j\omega n}| = 1$$

$X(e^{j\omega})$ converges if

$$\sum_{-\infty}^{+\infty} |x(n)| < \infty$$

stable system $\Leftarrow$ $H(e^{j\omega})$ converges

Example

① $x(n) = \left(\frac{1}{2}\right)^n u(n)$

$$\sum_{-\infty}^{+\infty} |x(n)| = 2$$

② $x(n) = (2)^n u(n)$

$$\sum_{-\infty}^{+\infty} |x(n)| = \infty$$

$$X_r(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} [x(n) r^{-n}] e^{-j\omega n}$$

$$= \sum_{-\infty}^{+\infty} x(n) \underbrace{(re^{j\omega})^{-n}}_{z}$$

The z-Transform

$$z = re^{j\omega}$$

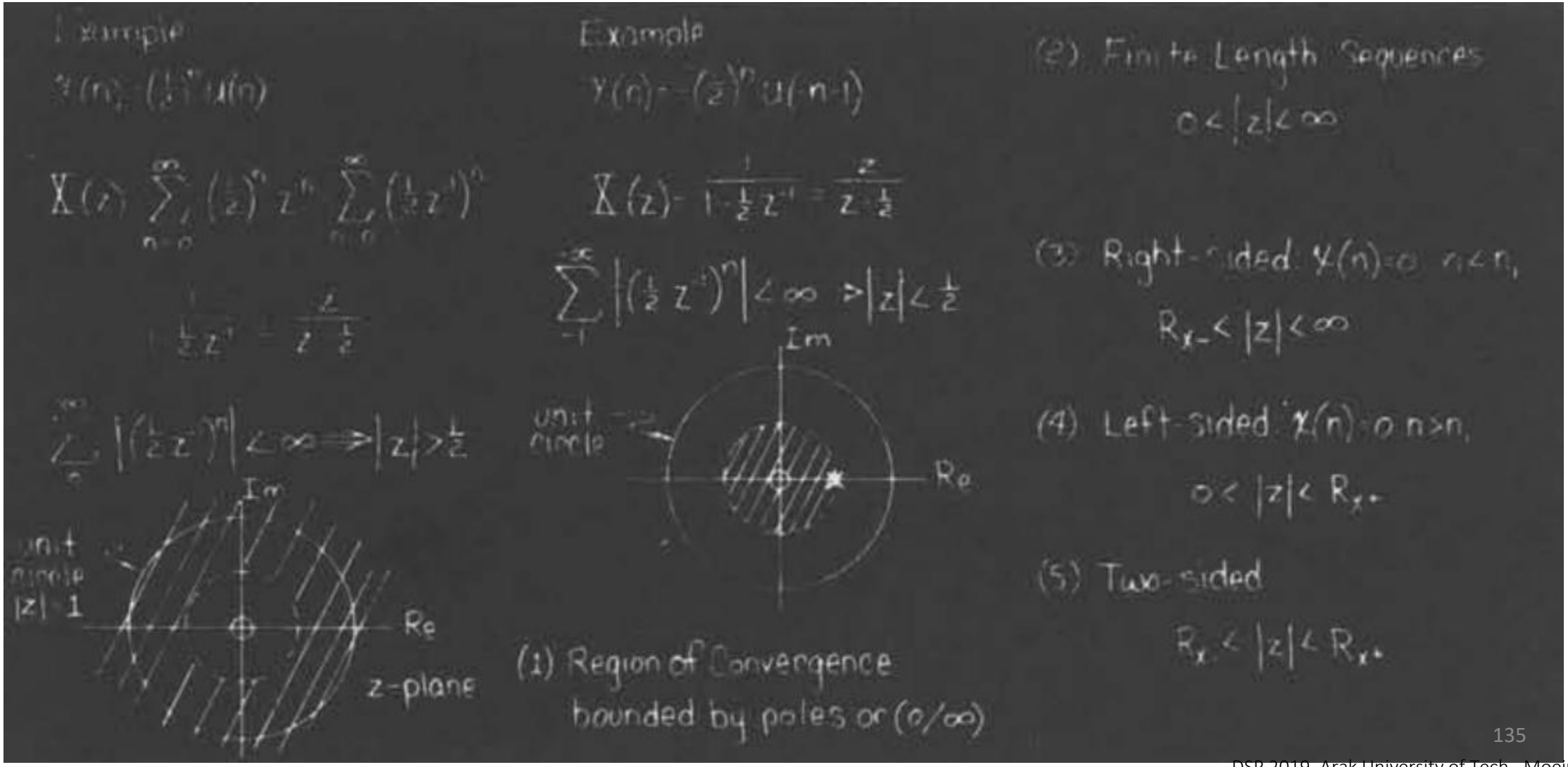
$$X(z) = \sum_{n=-\infty}^{+\infty} x(n) z^{-n}$$

$$X(e^{j\omega}) = X(z)\Big|_{z=e^{j\omega}}$$

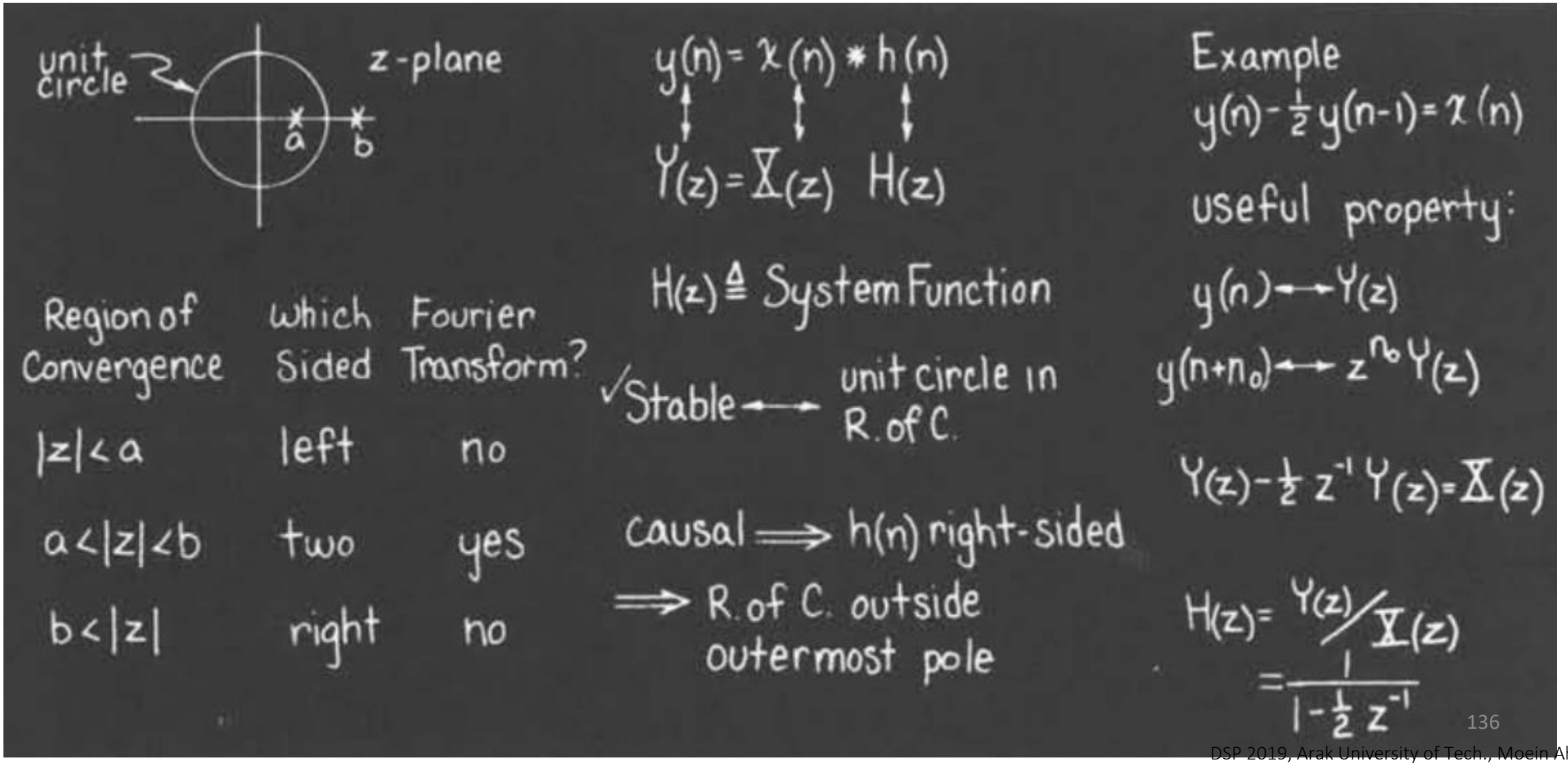
converges if

$$\sum_{n=-\infty}^{+\infty} |x(n) r^{-n}| < \infty$$

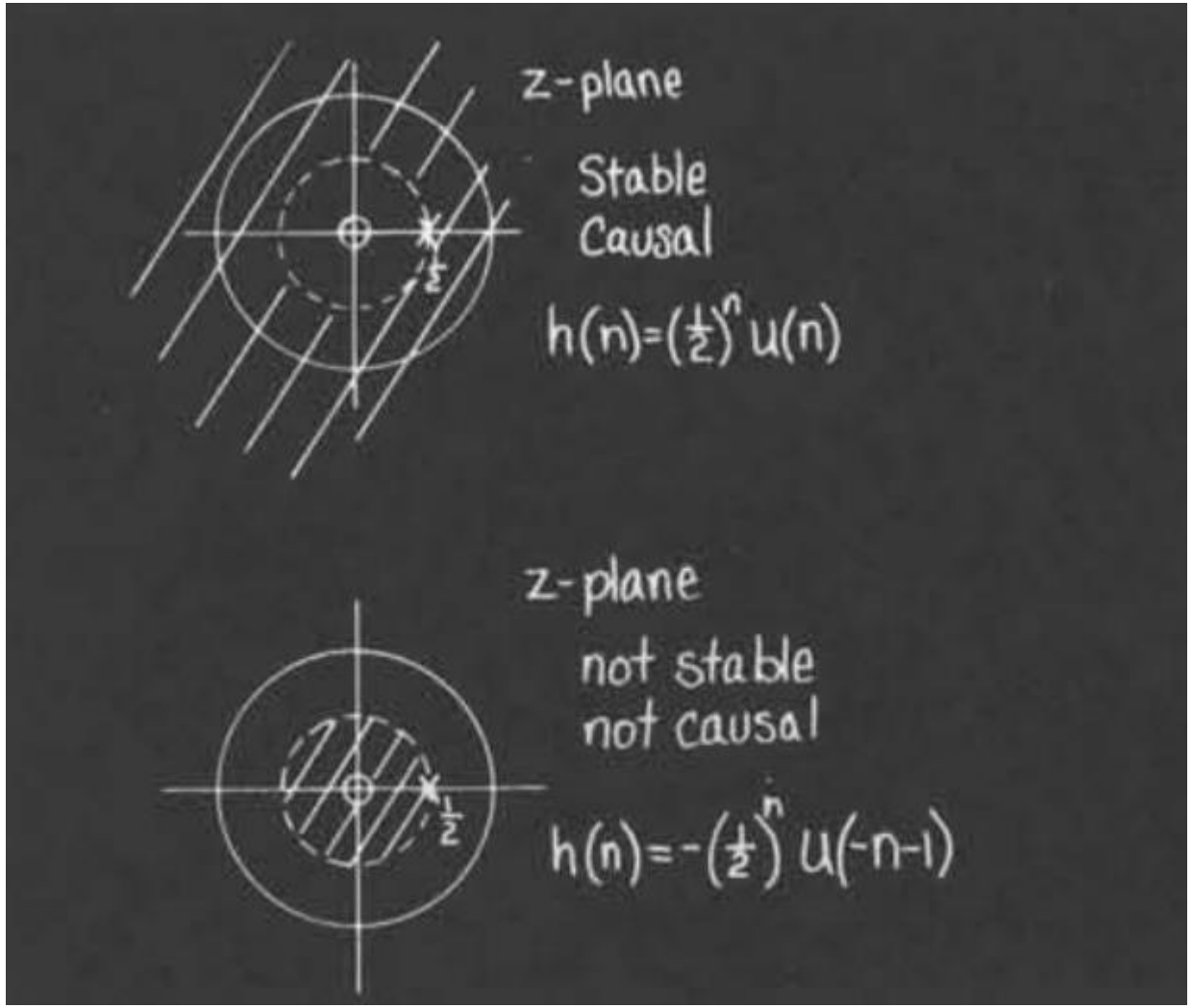
Class Review



Class Review



Class Review



Class Review

$$X(z)=\sum_{n=-\infty}^{+\infty} x(n)z^{-n}$$

Inverse z-transform

Inspection Method

$$a^n u(n) \longleftrightarrow \frac{1}{1-az^{-1}} \quad |z|>|a|$$

$$-a^n u(-n-1) \longleftrightarrow \frac{1}{1-az^{-1}} \quad |z|<|a|$$

ETC.

2. Power Series

$$X(z)=\frac{1}{1-az^{-1}}=\frac{z}{z-a}$$

$|z|>|a|$

$$1-az^{-1}\,\overline{)\,1}\quad 1+az^{-1}+a^2z^{-2}+a^3z^{-3}+\dots$$

$$\frac{1-az^{-1}}{az^{-1}}$$

$$\frac{az^{-1}-a^2z^{-2}}{a^2z^{-2}}$$

$|z|<|a|$

but also

$$\frac{1}{1-az^{-1}}=-a^{-1}z-a^{-2}z^2+\dots$$

3. Partial Fraction Exp.

$$F(x)=\frac{P(x)}{Q(x)}=\sum_{k=1}^{N}\frac{R_k}{x-x_k}$$

$$F(x)(x-x_r)\Big|_{(x=x_r)}$$

$$=\sum_{k=1}^{N}\underbrace{\frac{R_k}{(x-x_k)}(x-x_r)}\Big|_{(x=x_r)}$$

$$=0 \quad k\neq r$$
$$=R_r \quad k=r$$

$$R_r=F(x)(x-x_r)\Big|_{(x=x_r)}$$

= Residue of $F(x)$ at x_r

Class Review

$X=z \quad X(z)=\sum_k \frac{A_k}{z-a_k}$

$X=z^{-1} \quad X(z)=\sum_k \frac{B_k}{1-a_k z^{-1}}$

Example

$X(z)=\frac{1}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{4}z^{-1})} \quad |z|>\frac{1}{2}$

let $x \Rightarrow z^{-1}$

$$\frac{1}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{4}z^{-1})}=\frac{8}{(z^{-1}-2)(z^{-1}-4)}$$

$$\frac{8}{(z^{-1}-2)(z^{-1}-4)}(z^{-1}-2)\Big|_{z^{-1}=2}=-4$$

$$\frac{8}{(z^{-1}-2)(z^{-1}-4)}(z^{-1}-4)\Big|_{z^{-1}=4}=4$$

$$X(z)=\frac{-4}{z^{-1}-2}+\frac{4}{z^{-1}-4}$$

$$=\underbrace{\frac{2}{1-\frac{1}{2}z^{-1}}}+\underbrace{\frac{-1}{1-\frac{1}{4}z^{-1}}}$$

$$2(\tfrac{1}{2})^n u(n)-(\tfrac{1}{4})^n u(n)$$

4. Contour Integration

$$x(n)=\frac{1}{2\pi j}\oint_C X(z)\,z^{n-1}\,dz$$

$=\sum$ (residues of $X(z)\,z^{n-1}$ at poles inside C)

Example

$$X(z)=\frac{1}{1-\frac{1}{2}z^{-1}}$$

$$=\frac{z}{z-\frac{1}{2}} \quad |z|>\frac{1}{2}$$

Class Review

unit circle

z-plane

$n \geq 0 \quad x(n) = \left(\frac{1}{2}\right)^n$

$n < 0$ 1 pole at $z = \frac{1}{2}$

n poles at $z = 0$

$x(n) = \frac{-1}{2\pi j} \oint_{c'} X\left(\frac{1}{p}\right) p^{-n-1} dp$

$= \frac{1}{2\pi j} \oint_{c'} X\left(\frac{1}{p}\right) p^{-n-1} dp$

$X(z) z^{n-1} = z^n / \left(z - \frac{1}{2}\right)$

$n \geq 0$ 1 pole at $z = \frac{1}{2}$

Residue of $\frac{z^n}{z - \frac{1}{2}}$ at $z = \frac{1}{2}$

$\left.\frac{z^n}{(z - \frac{1}{2})}(z - \frac{1}{2})\right|_{z=\frac{1}{2}} = \left(\frac{1}{2}\right)^n$

Easy way:

$x(n) = \frac{1}{2\pi j} \oint_c X(z) z^{n-1} dz$

let $z = p^{-1} \quad z^{n-1} = p^{-n+1}$

If $z = re^{j\theta}$

then $p = (1/r) e^{-j\theta}$

Class Review

Previous Example

$$X(z)=\frac{1}{1-\frac{1}{2}z^{-1}} \quad |z|>\frac{1}{2}$$

$$X(1/p)=\frac{1}{1-\frac{1}{2}p} \quad |p|<2$$

$$=\frac{-2}{p-2}$$

$$X(1/p)\ p^{-n-1}$$

$$x(n)=\left(\tfrac{1}{2}\right)^n u(n)$$

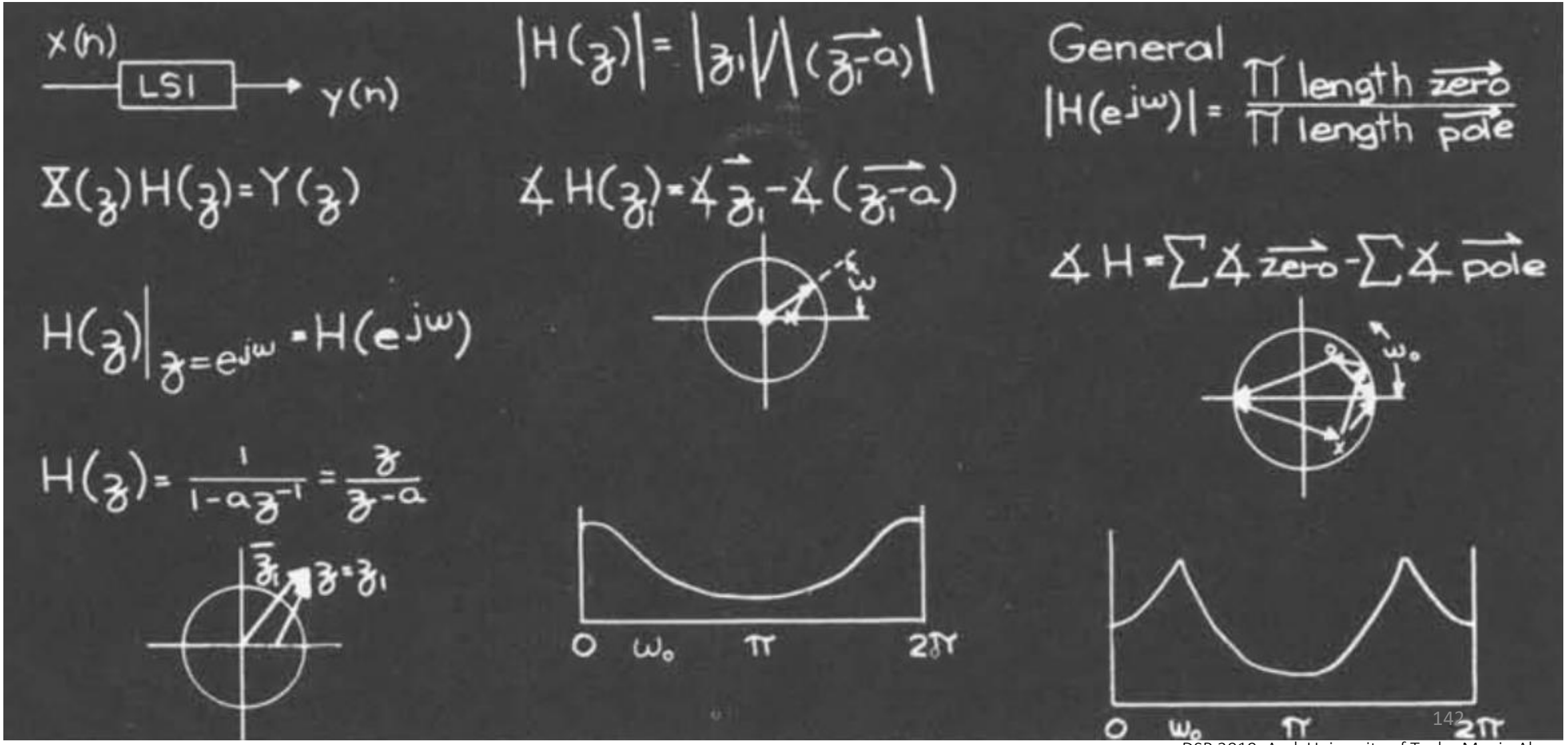
$n<0$ 1 pole at $p=2$

$\therefore x(n)=0 \quad n<0$

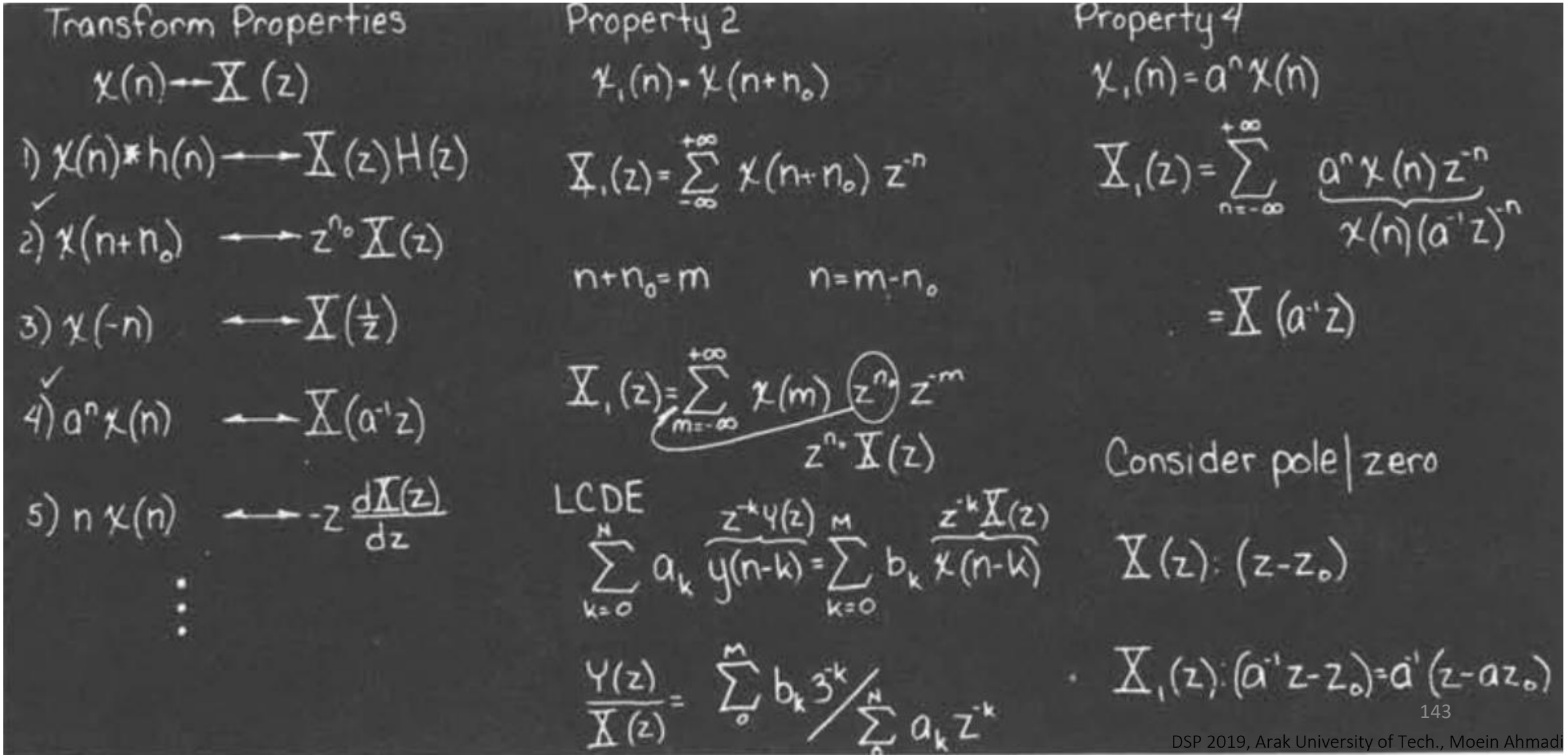
$n\geq 0$ 1 pole at $p=2$

$(n+1)$ poles at $p=0$

Class Review



Class Review



Class Review

pole(zero) pole (zero)

$z_0 \longrightarrow az_0$

$r_0 e^{j\theta_0} \longrightarrow |a| r_0 e^{j(\theta_0 + \theta_n)}$

Boxcar Sequence

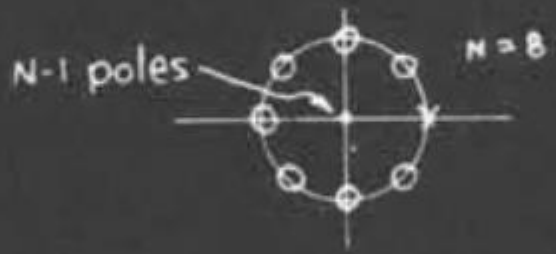
$x(n) = 1 \quad 0 \le n \le (N-1)$

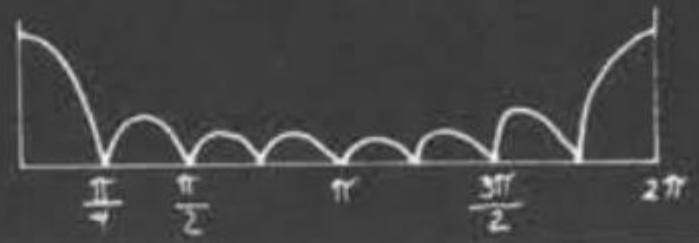
$= 0$ otherwise

$x(n) = u(n) - u(n-N)$

$$X(z) = \frac{1}{1-z^{-1}} - \frac{z^{-N}}{1-z^{-1}}$$

$$= \frac{1-z^{-N}}{1-z^{-1}} = \frac{z^N - 1}{z^{(N-1)}(z-1)}$$





$$X(e^{j\omega}) = \frac{1-e^{-j\omega N}}{1-e^{-j\omega}}$$

$$= \frac{e^{-j\omega\frac{N}{2}}\overbrace{\left[e^{j\omega\frac{N}{2}} - e^{-j\omega\frac{N}{2}}\right]}^{2j\sin\frac{\omega N}{2}}}{e^{-j\frac{\omega}{2}}\underbrace{\left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}\right]}_{2j\sin\frac{\omega}{2}}}$$

$$= e^{j\omega\frac{(N-1)}{2}} \frac{\sin(\omega\frac{N}{2})}{\sin(\frac{\omega}{2})}$$

Class Review

pole(zero) pole (zero)

$z_0 \longrightarrow az_0$

$r_0 e^{j\theta_0} \longrightarrow |a| r_0 e^{j(\theta_0 + \theta_n)}$

Boxcar Sequence

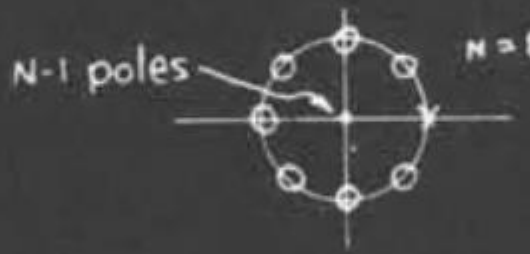
$x(n) = 1 \quad 0 \le n \le (N-1)$

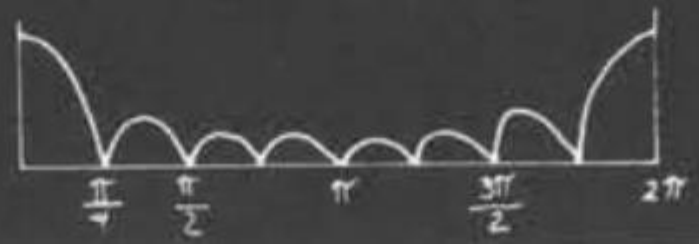
$= 0$ otherwise

$x(n) = u(n) - u(n-N)$

$$X(z) = \frac{1}{1-z^{-1}} - \frac{z^{-N}}{1-z^{-1}}$$

$$= \frac{1-z^{-N}}{1-z^{-1}} = \frac{z^N - 1}{z^{(N-1)}(z-1)}$$



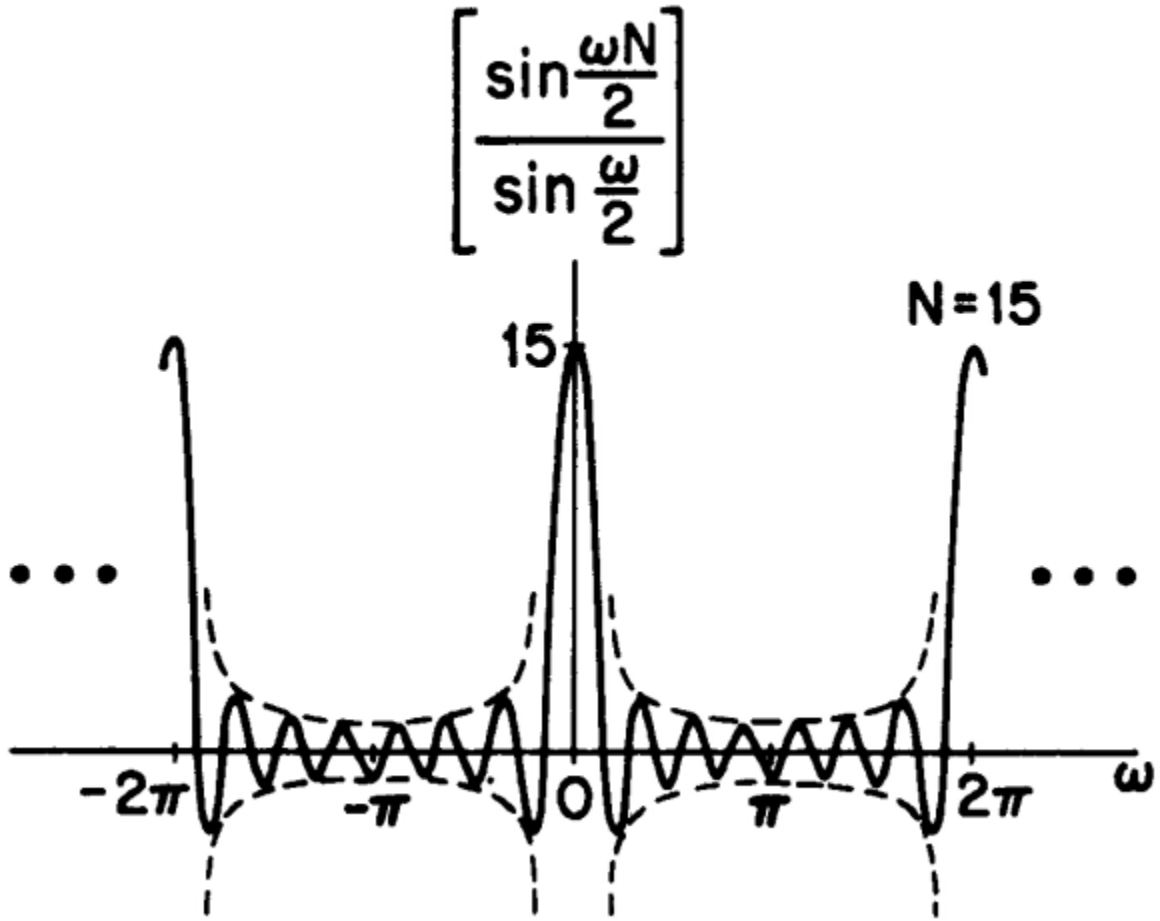


$$X(e^{j\omega}) = \frac{1-e^{-j\omega N}}{1-e^{-j\omega}}$$

$$= \frac{e^{-j\omega\frac{N}{2}} \overbrace{\left[e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}}\right]}^{2j \sin\frac{\omega N}{2}}}{e^{-j\frac{\omega}{2}} \underbrace{\left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}\right]}_{2j \sin\frac{\omega}{2}}}$$

$$= e^{j\omega\frac{(N-1)}{2}} \frac{\sin(\omega\frac{N}{2})}{\sin(\frac{\omega}{2})}$$

Class Review



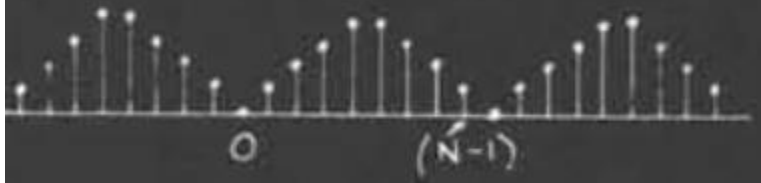
Fourier Transform of a rectangular sequence.

Class Review

$x(n)$ Finite length

$\tilde{x}(n)$ Periodic

$\tilde{x}(n) = x(n) + x(n+N) + \dots$



$x(n) = \tilde{x}(n) \quad 0 \le n \le (N-1)$

$= 0 \quad$ otherwise

$x(n) = \tilde{x}(n)\, R_N(n)$

$R_N(n) = 1 \quad 0 \le n \le (N-1)$

$= 0 \quad$ otherwise

$\tilde{x}(n)$ has a Fourier Series Representation

DFS of $\tilde{x}(n)$ $\triangleq$ DFT of $x(n)$

Discrete Fourier Series

$\tilde{x}(n)$: periodic, period N

$$\tilde{x}(n) = \sum \tilde{X}(k) e^{j \frac{2\pi}{N} nk}$$

Class Review

$$e^{j\frac{2\pi}{N}nk} = \underbrace{e^{j\frac{2\pi}{N}n(k+N)}}_{e^{j\frac{2\pi}{N}nk}\,e^{j\frac{2\pi}{N}nN}\,(1)}$$

$$\tilde{x}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}(k)e^{j\frac{2\pi}{N}nk}$$

$$\tilde{X}(k) = \sum_{n=0}^{N-1}\tilde{x}(n)\,e^{-j\frac{2\pi}{N}nk}$$

$$e^{-j\frac{2\pi}{N}n(k+N)} = e^{-j\frac{2\pi}{N}nk}$$

$\tilde{X}(k)$ periodic in k

period N

$$W_N \triangleq e^{-j\frac{2\pi}{N}}$$

$$\tilde{x}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}(k)\,W_N^{-nk}$$

$$\tilde{X}(k) = \sum_{n=0}^{N-1}\tilde{x}(n)\,W_N^{nk}$$

DFS Properties

Shifting

$\tilde{x}(n+m)$ $\quad W_N^{-km}\tilde{X}(k)$

$W_N^{\ell n}\tilde{x}(n)$ $\quad \tilde{X}(k+\ell)$

Symmetry: $\tilde{x}(n)$ real

$$\tilde{X}(k) = \tilde{X}_R(k) + j\tilde{X}_I(k)$$

$$\tilde{X}_R(k) = \tilde{X}_R(-k) \quad \text{even}$$
$$= \tilde{X}_R(N-k)$$

$$\tilde{X}_I(k) = -\tilde{X}_I(-k) \quad \text{odd}$$
$$= -\tilde{X}_I(N-k)$$

$|\tilde{X}(k)|$ even

$\measuredangle X(k)$ odd

Class Review

Convolution Property

$$\tilde{x}_1(n) \leftrightarrow \tilde{X}_1(k)$$

$$\tilde{x}_2(n) \leftrightarrow \tilde{X}_2(k)$$

$$\tilde{x}_3(n) \leftrightarrow \tilde{X}_1(k)\,\tilde{X}_2(k)$$

$$\tilde{x}_3(n) = \sum_{m=0}^{N-1} \tilde{x}_1(m)\,\tilde{x}_2(n-m)$$

Dual Property

$$\tilde{x}_4(n) = \tilde{x}_1(n)\,\tilde{x}_2(n)$$

$$\tilde{X}_4(k) = \frac{1}{N} \sum_{\ell=0}^{N-1} \tilde{X}_1(\ell)\,\tilde{X}_2(k-\ell)$$

Class Review

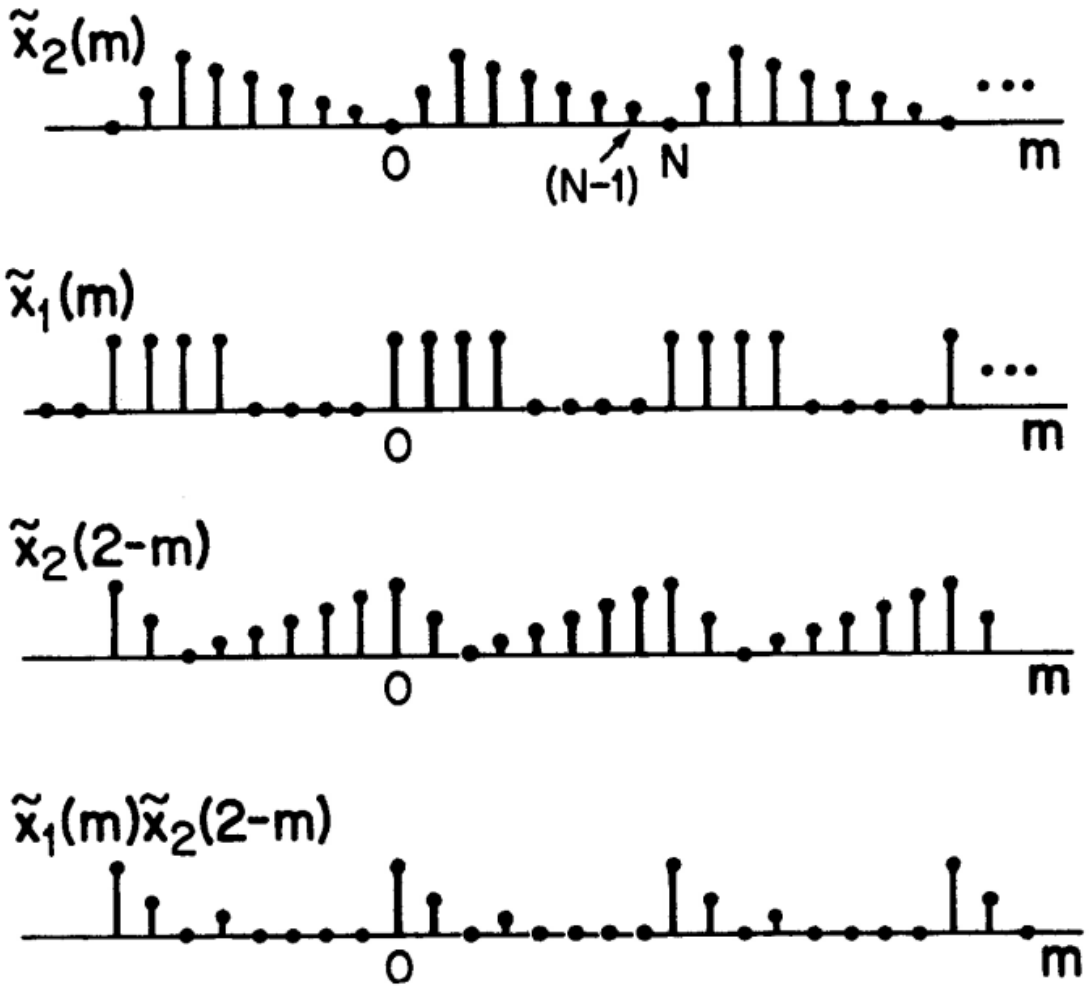


Illustration of the sequences involved in forming a periodic convolution.

Class Review

$x(n)=0 \quad n<0,\ n>(N-1)$

finite length N (or less)

$\tilde{x}(n)=\sum_{r=-\infty}^{+\infty} x(n+rN)$

$= x(n \text{ modulo } N) \triangleq x((n))_N$

$x(n)=\tilde{x}(n)\, R_N(n)$

$\tilde{X}(k)=$ DFS of $\tilde{x}(n)$

Discrete Fourier Series

$\tilde{X}(k)=\sum_{n=0}^{N-1} \tilde{x}(n) W_N^{nk}$

$\tilde{x}(n)=\frac{1}{N}\sum_{k=0}^{N-1} \tilde{X}(k) W_N^{-nk}$

Discrete Fourier Transform

$X(k)=\sum_{n=0}^{N-1} x(n) W_N^{nk} \quad k=0,1,\dots,N-1$

$=0$ otherwise

$X(k)=\tilde{X}(k)\, R_N(k)$

$\tilde{X}(k)=X((k))_N$

$X(k)=\left[\sum_{n=0}^{N-1} x(n) W_N^{nk}\right] R_N(k)$

$x(n)=\left[\frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-nk}\right] R_N(n)$

$X(z)=\sum_{n=0}^{N-1} x(n) z^{-n} \quad k=0,1,\dots,N-1$

$X(k)=X(z)\big|_{z=W_N^{-k}}$

$N=8$

Class Review

Properties of the DFT

Shifting Property

$x(n) \longleftrightarrow X(k)$

$\tilde{x}(n) \longleftrightarrow \tilde{X}(k)$

$\tilde{x}_1(n) = \tilde{x}(n+m) \longleftrightarrow \tilde{X}(k) W_N^{-km}$

$x_1(n) \longleftrightarrow X(k) W_N^{-km}$

$x((n+m))_N R_N(n) \longleftrightarrow W_N^{-km} X(k)$

$W_N^{\ell n} x(n) \longleftrightarrow X((k+\ell))_N R_N(k)$

Symmetry Properties

DFS $\tilde{x}(n)$ real

$\tilde{X}_R(k) = \tilde{X}_R(N-k)$

$\tilde{X}_I(k) = -\tilde{X}_I(N-k)$

DFT $x(n)$ real

$X_R(k) = X_R((N-k))_N R_N(k)$ (even)

$X_I(k) = -X_I((N-k))_N R_N(k)$ (odd)

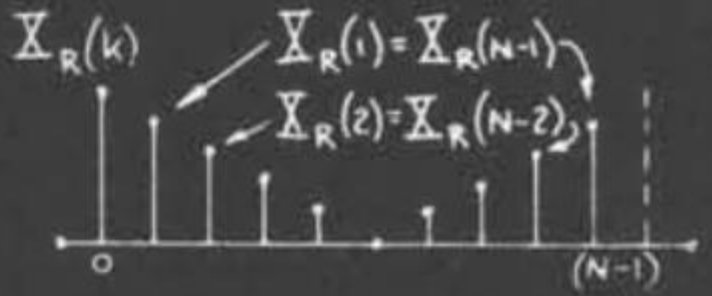
for example

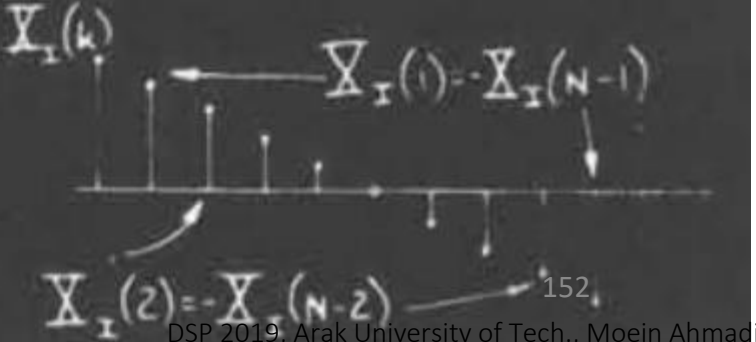
$X_R(0)$

$X_R(0) = X_R((N-0))_N R_N(0)$ (1)

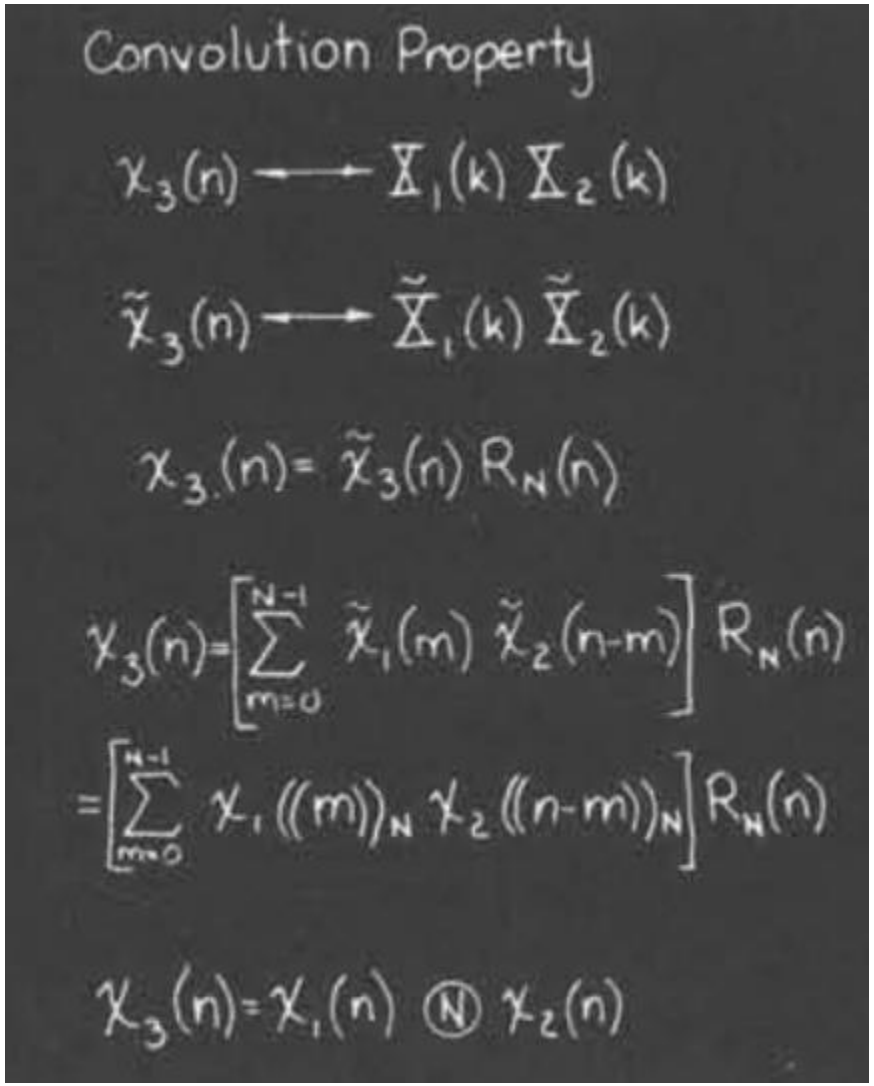
$X_R(N-1)$

$X_R(1) = X_R((N-1))_N R_N(1)$ (1)

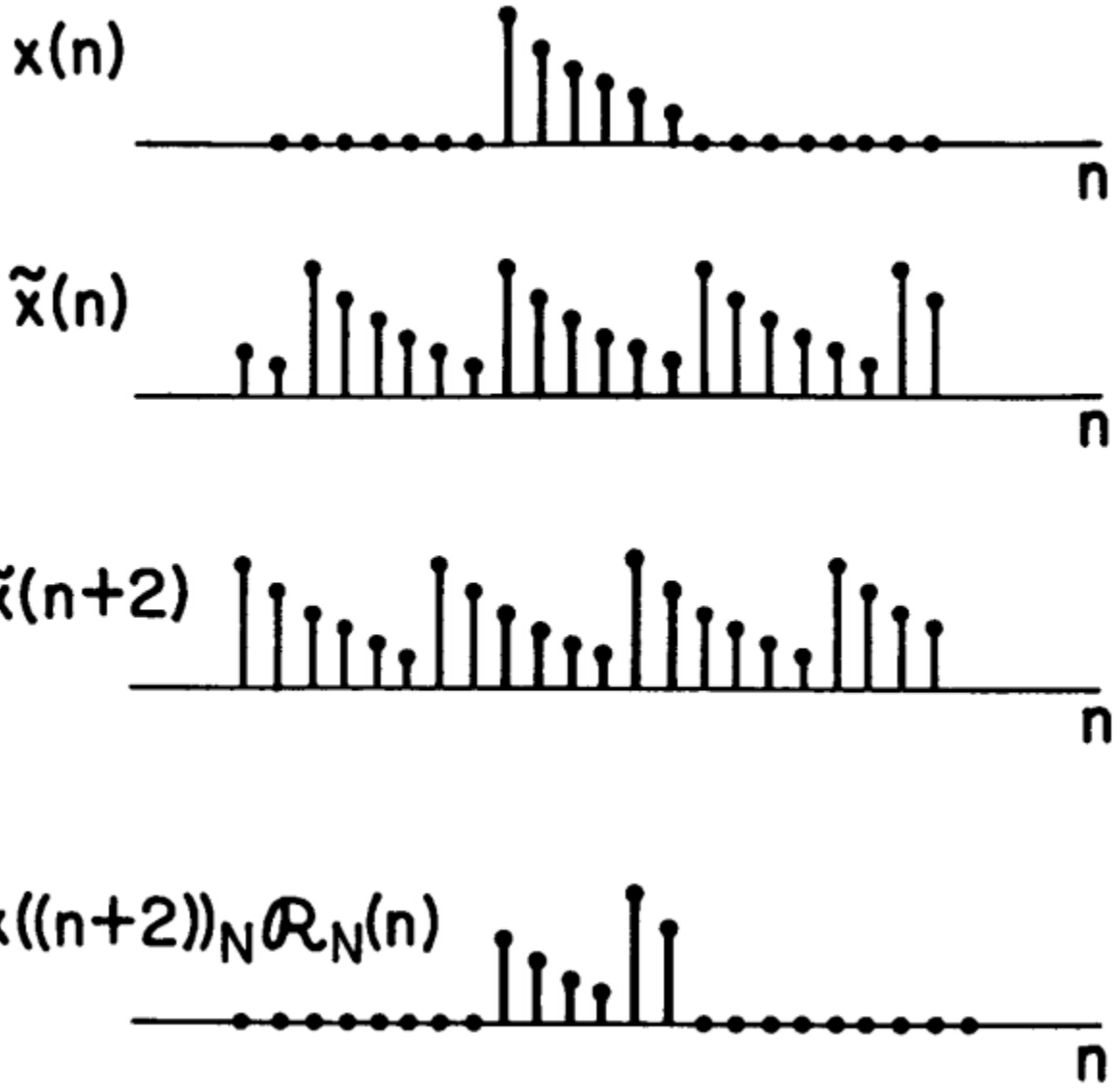




Class Review



Class Review



Circular shifting of a finite length sequence.

Class Review

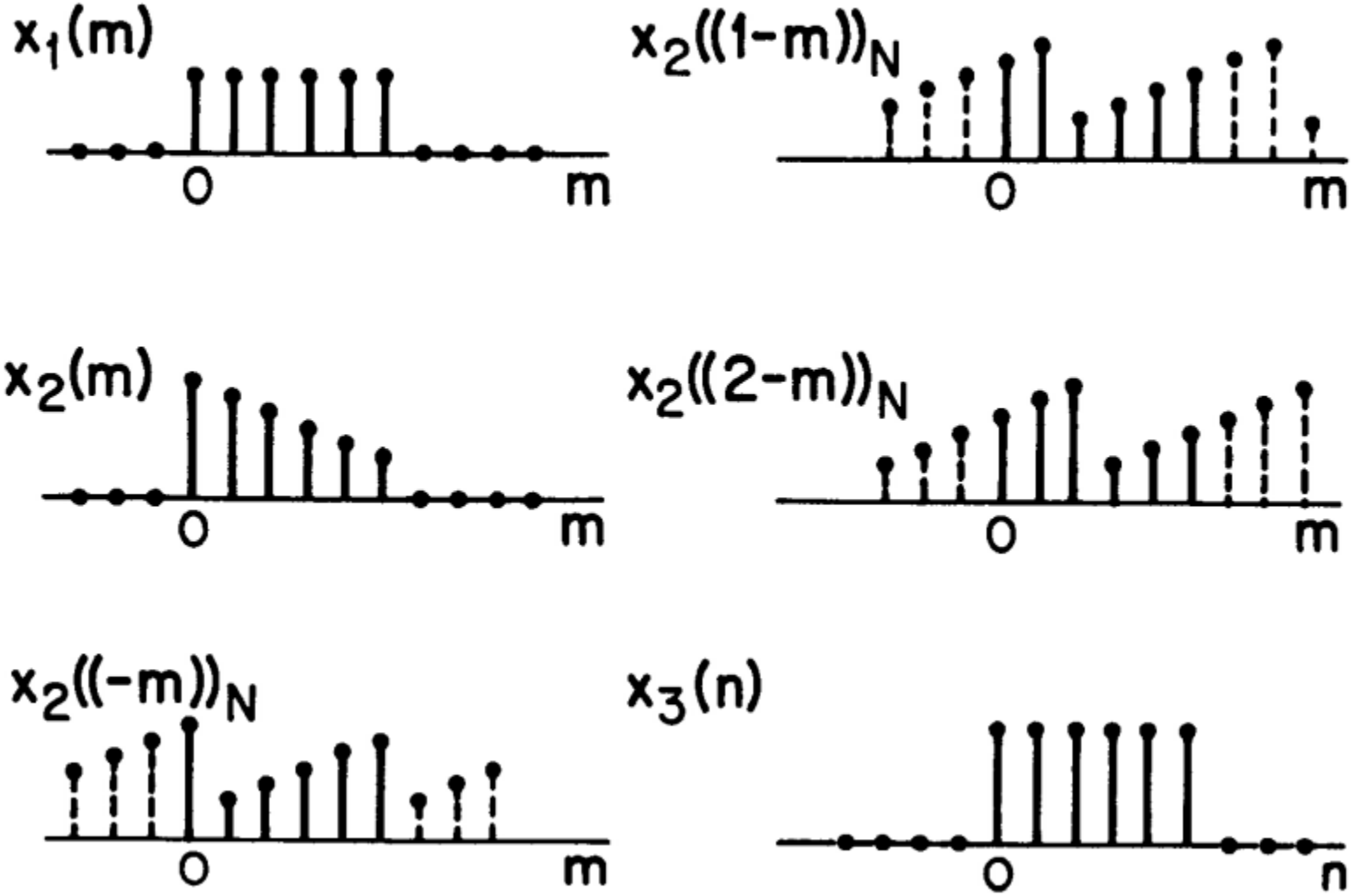


Illustration of circular convolution. (Note that $x_2((-m))_N$ is incorrectly drawn. In Problem 9.4 you are asked to correct this.)

Class Review

Circular Convolution $x_3(n) = x_1(n) \textcircled{N} x_2(n)$

$$= \left[\sum_{m=0}^{N-1} \tilde{x}_1(m)\, \tilde{x}_2(n-m)\right] \mathcal{R}_N(n)$$

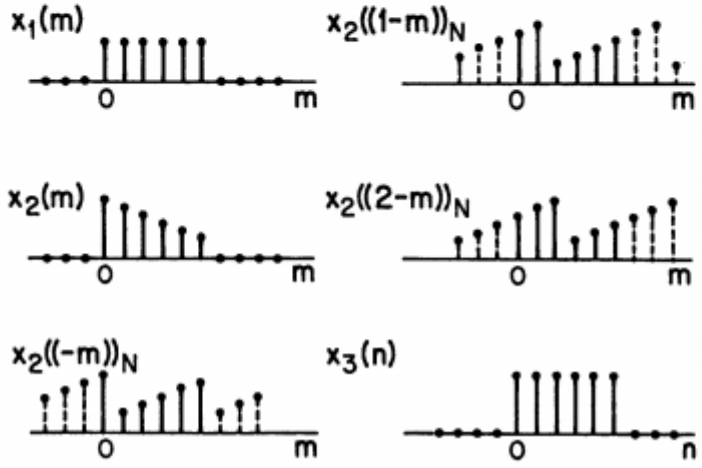
$$= \left[\sum_{m=0}^{N-1} x_1((m))_N\, x_2((n-m))_N\right] \mathcal{R}_N(n)$$

$$= \left[\sum_{m=0}^{N-1} x_1(m)\, x_2((n-m))_N\right] \mathcal{R}_N(n)$$

$$= \left[x_1(n) * x_2((n))_N\right] \mathcal{R}_N(n)$$

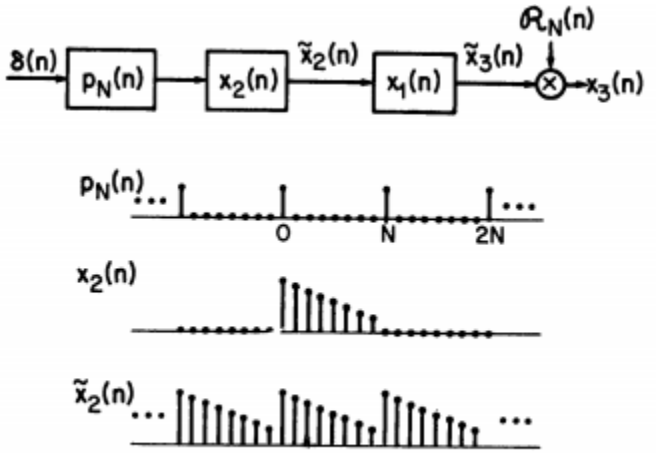
a.

Circular convolution expressed in terms of periodic and linear convolution.



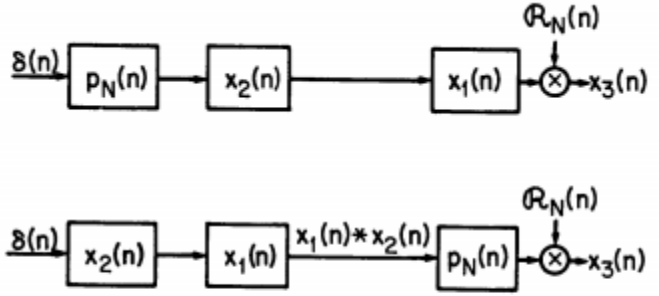
b.

Example of circular convolution of two sequences.



c.

An interpretation of circular convolution.



d.

Rearrangement of the operations in forming the circular convolution.

Class Review

"Circular Convolution =
Linear Convolution + Aliasing"

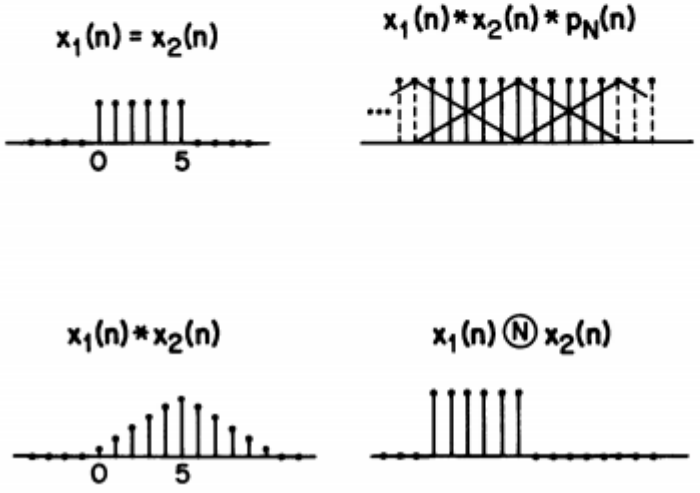
$$\hat{x}_3(n) = x_1(n) * x_2(n)$$

$$x_3(n) = x_1(n) \textcircled{N} x_2(n)$$

$$x_3(n) = \left[\sum_{r=-\infty}^{+\infty} \hat{x}_3(n+rN)\right] R_N(n)$$

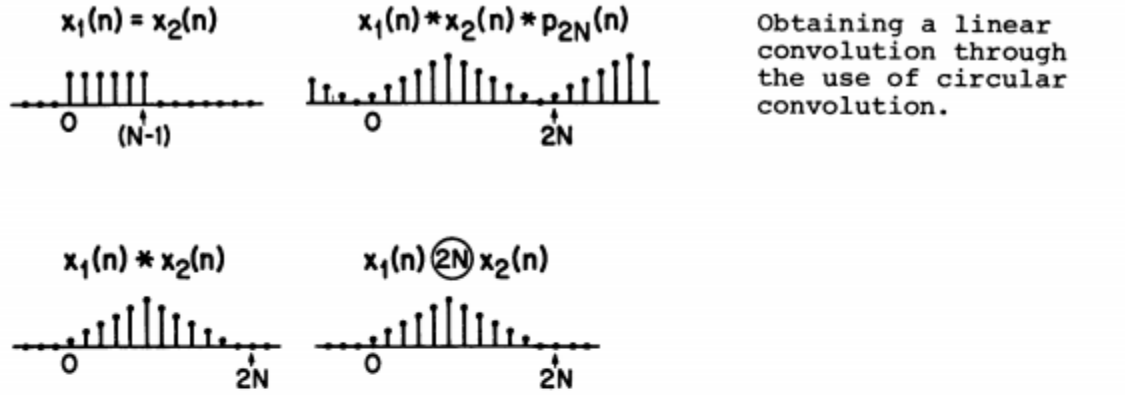
e.

Interpretation of circular convolution as linear convolution followed by aliasing.



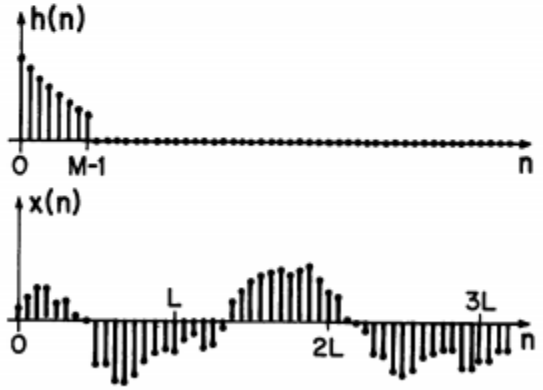
f.

Example of a circular convolution formed by linear convolution followed by aliasing.



g.

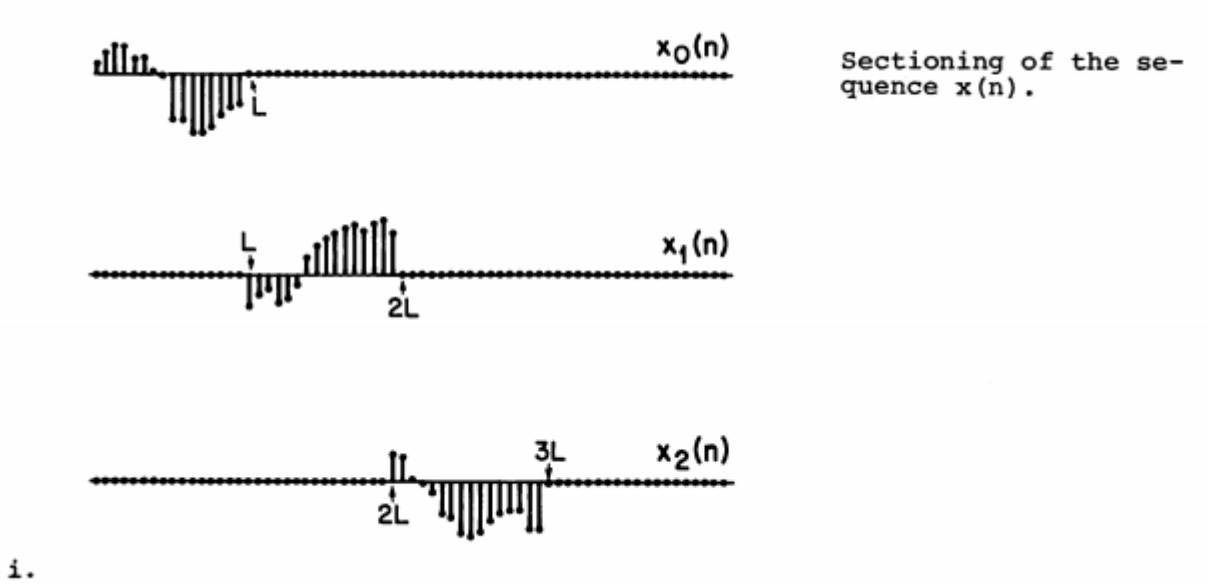
Obtaining a linear convolution through the use of circular convolution.



h.

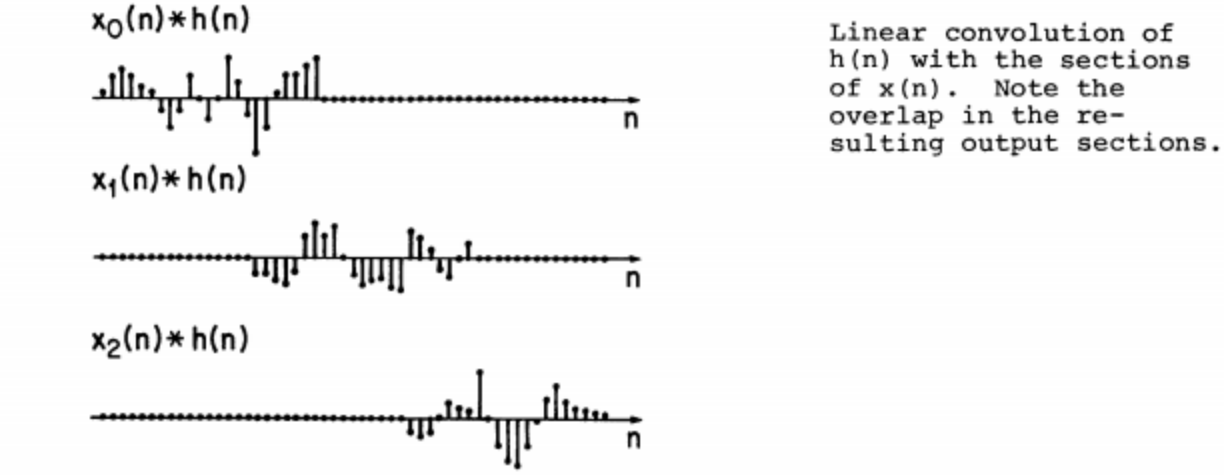
A finite length unit sample response and a sequence of indefinite length.

Class Review



Sectioning of the sequence x(n).

i.



Linear convolution of h(n) with the sections of x(n). Note the overlap in the resulting output sections.

j.

Class Review

Digital Networks

N^{th} order Difference Eq

$$y(n) = \sum_{k=1}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$$

$$y(n) = \sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

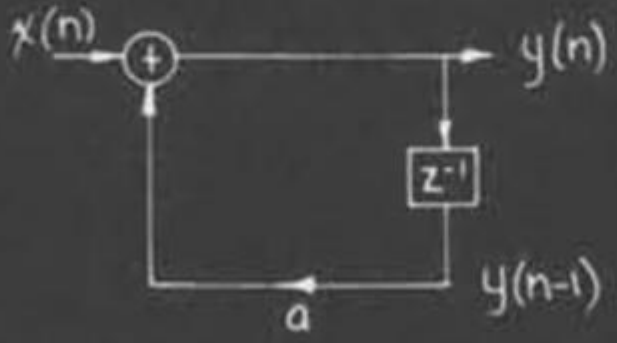
Basic Operations

delay $x(n) \rightarrow [z^{-1}] \rightarrow x(n-1)$, $X(z) \rightarrow z^{-1}X(z)$

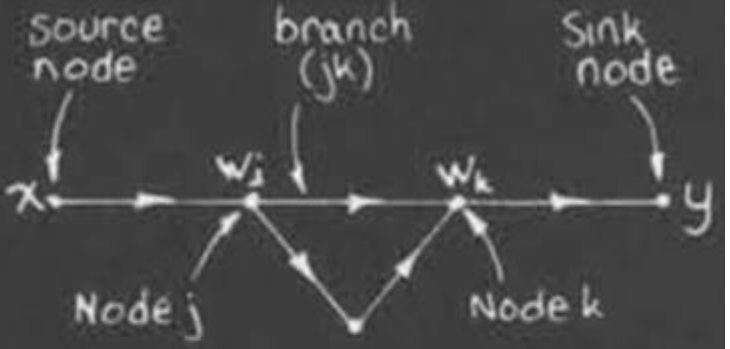
Multiplication $x(n) \xrightarrow{a} ax(n)$

Addition $x_1 \rightarrow (+) \rightarrow x_1 + x_2$, x_2

Example: $y(n) = ay(n-1) + x(n)$



Signal Flow Graph



branch (jk): input = w_j

output $\triangleq v_{jk} = f_{jk}(w_j)$

Node value = $\sum$ outputs of entering branches

S_{jk} j^{th} source to k^{th} node

Class Review

$$W_k = \underbrace{\sum_{j=1}^{N} r_{jk}}_{\text{Network}} + \underbrace{\sum_{j=1}^{M} S_{jk}}_{\text{Source}}$$

Linear Signal Flow Graph

arbitrary branch (jk)

$$r_{jk} = t_{jk} W_j$$

$$y(n) = a\,y(n-1) + x(n)$$

Matrix Representation of Digital Networks

$$W_k(z) = \underbrace{\sum_{j=1}^{N} V_{jk}(z)}_{\text{Network}} + \underbrace{\sum_{j=1}^{M} S_{jk}(z)}_{\text{Source.}}$$

$$V_{jk}(z) = F_{jk}(z)\,W_j(z)$$

$$S_{jk}(z) = b_{jk}\,X_j(z)$$

$$W_k(z) = \sum_{j=1}^{N} F_{jk}(z)\,W_j(z) + \sum_{j=1}^{M} b_{jk}\,X_j(z)$$

$$\begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_N \end{bmatrix} = \begin{bmatrix} F_{11} & F_{21} & \cdots & F_{N1} \\ F_{12} & & & \\ \vdots & & & \\ F_{1N} & & & F_{NN} \end{bmatrix} \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_N \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{M1} \\ \vdots & & \\ b_{1N} & \cdots & b_{MN} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_M \end{bmatrix}$$

Class Review

$$\underline{W}(z)=\underline{F}^{t}(z)\underline{W}(z)+\underline{B}^{t}\underline{X}(z)$$

$$F(z)=\{F_{kj}(z)\}$$

$$\underline{F}^{t}(z)=\underline{F}_{c}^{t}+z^{-1}\underline{F}_{d}^{t}$$

$$\underline{W}(z)=\underline{F}_{c}^{t}\underline{W}(z)+z^{-1}\underline{F}_{d}^{t}\underline{W}(z)+\underline{B}^{t}\underline{X}(z)$$

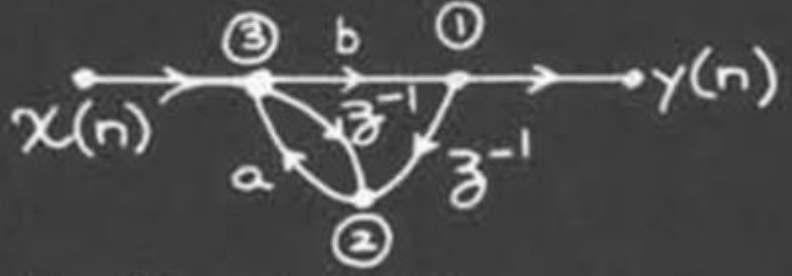
$$\underline{Y}(z)=\underline{C}^{t}\underline{W}(z)$$

or:

$$\underline{W}(n)=\underline{F}_{c}^{t}\underline{W}(n)+\underline{F}_{d}^{t}\underline{W}(n-1)+\underline{B}^{t}\underline{X}(n)$$

$$y(n)=\underline{C}^{t}\underline{W}(n)$$

Example

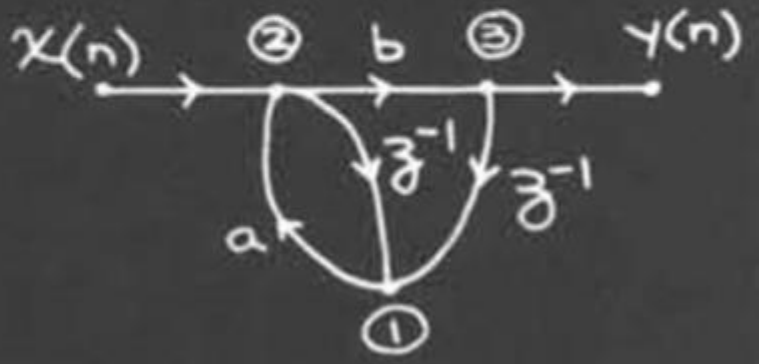


$$W_1(n)=bW_3(n)$$

$$W_2(n)=W_1(n-1)+W_3(n-1)$$

$$W_3(n)=aW_2(n)+\chi(n)$$

$$\begin{bmatrix}W_1\\W_2\\W_3\end{bmatrix}=\begin{bmatrix}0&0&b\\0&0&0\\0&a&0\end{bmatrix}\begin{bmatrix}W_1(n)\\W_2(n)\\W_3(n)\end{bmatrix}+\begin{bmatrix}0&0&0\\1&0&1\\0&0&0\end{bmatrix}\begin{bmatrix}W_1(n-1)\\W_2(n-1)\\W_3(n-1)\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}\chi(n)$$



Class Review

$$\begin{bmatrix} w_1(n) \\ w_2(n) \\ w_3(n) \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ a & a & 0 \\ 0 & b & a \end{bmatrix} \begin{bmatrix} w_1(n) \\ w_2(n) \\ w_3(n) \end{bmatrix}$$

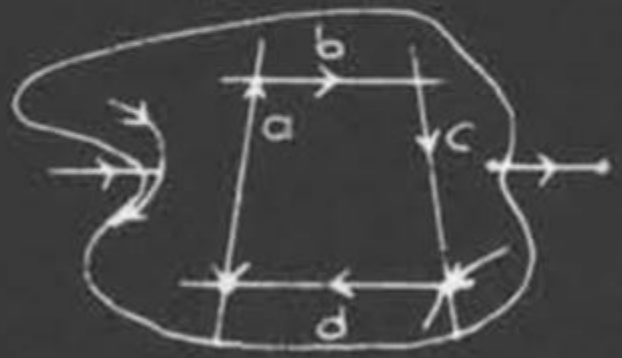
$$+ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1(n-1) \\ w_2(n-1) \\ w_3(n-1) \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \chi(n)$$

Network Computable

Nodes can be numbered so that F_c^t is

$$F_c^t = \begin{bmatrix} 0 & & \\ & 0 & \\ & & 0 \end{bmatrix}$$



Non computable Network

Class Review

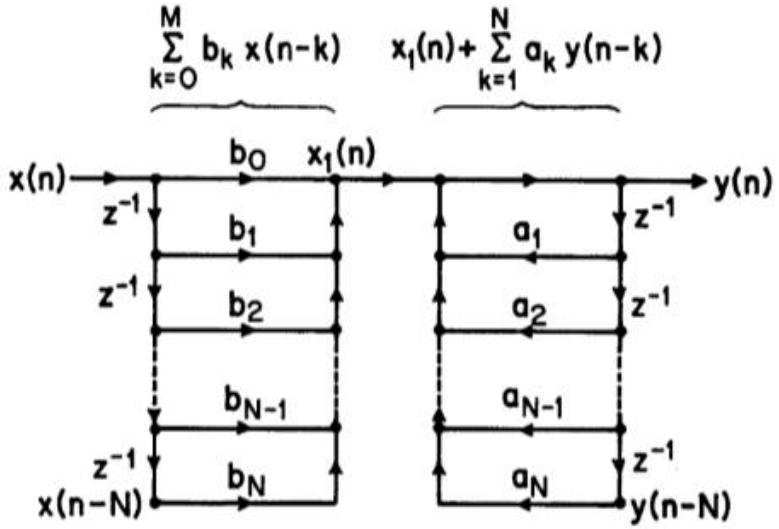
$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) + \sum_{k=1}^{N} a_k y(n-k)$$

$$H(z) = \left[\sum_{k=0}^{M} b_k z^{-k}\right]\left[\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}\right]$$

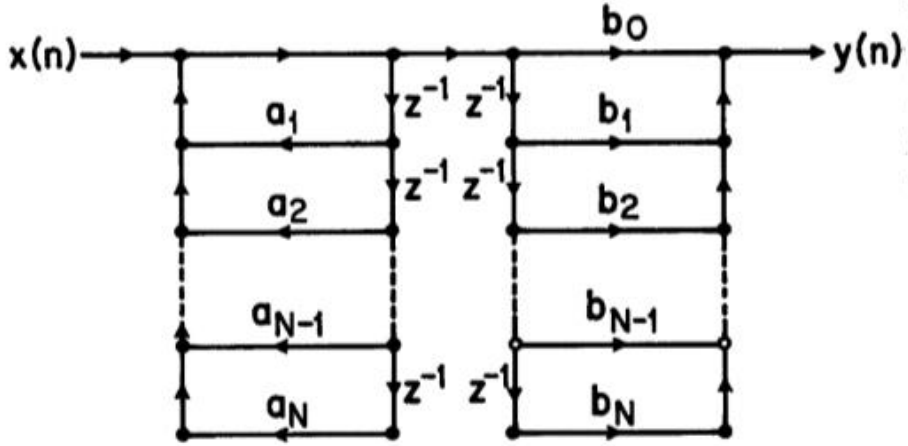
z-transform and difference equation for a general IIR system.

a.



Flow-graph representation of a general difference equation based on the factorization in b. (Direct form I realization.)

Class Review



Flow-graph representation of a general difference equation based on interchanging the order in which the poles and zeros are cascaded.

$$y(n) = \sum_{k=0}^{M} b_k\, x(n-k) + \sum_{k=1}^{N} a_k\, y(n-k)$$

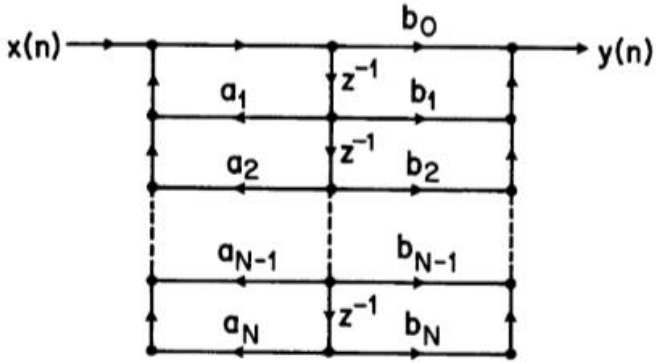
z-tranform factorization and difference equation corresponding to the network in c.

$$H(z) = \left[\frac{1}{1-\sum_{k=1}^{N} a_k z^{-k}}\right]\left[\sum_{k=0}^{M} b_k z^{-k}\right]$$

$$y_1(n) = x(n) + \sum_{k=1}^{N} a_k\, y_1(n-k)$$

$$y(n) = \sum_{k=0}^{M} b_k\, y_1(n-k)$$

Class Review



Flowgraph of c. collapsed to share delays (direct form II realization.)

TRANSPOSITION THEOREM

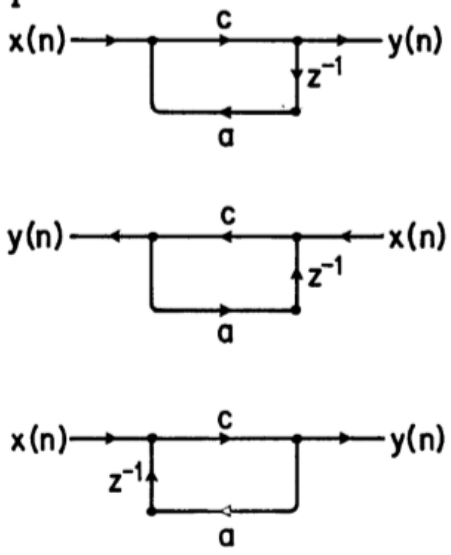
1. REVERSE DIRECTION OF ALL BRANCHES
2. INTERCHANGE INPUT AND OUTPUT

TRANSFER FUNCTION REMAINS THE SAME

Transposition theorem for signal flow-graphs.

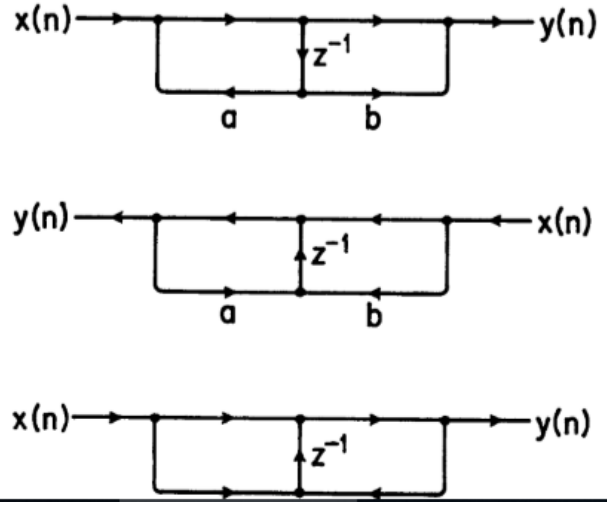
Class Review

Example 1



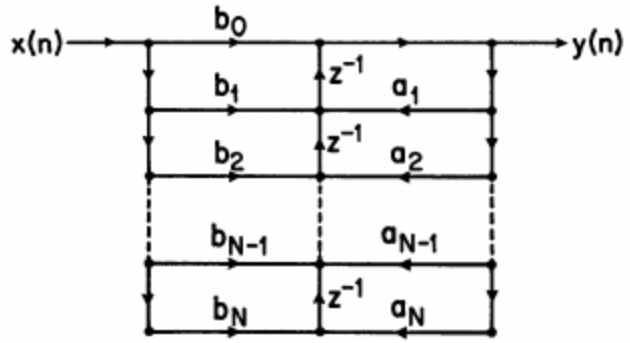
Example of Transposition theorem.

Example 2



Example of Transposition theorem.

Class Review



Transposed direct form II structure.

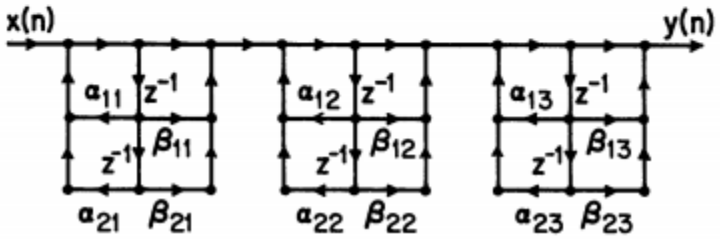
Cascade Structure

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

$$= A \prod \frac{1+\beta_{1k} z^{-1} + \beta_{2k} z^{-2}}{1 - \alpha_{1k} z^{-1} - \alpha_{2k} z^{-2}}$$

Factorization of the z-transform for the cascade structure.

j.



Cascade structure with a direct form II realization of each second-order subsystem.

Class Review

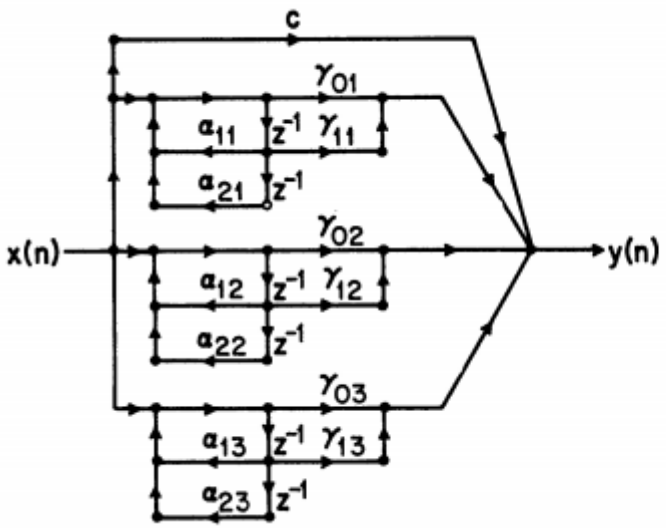
k.

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

Partial Fraction expansion for parallel structure.

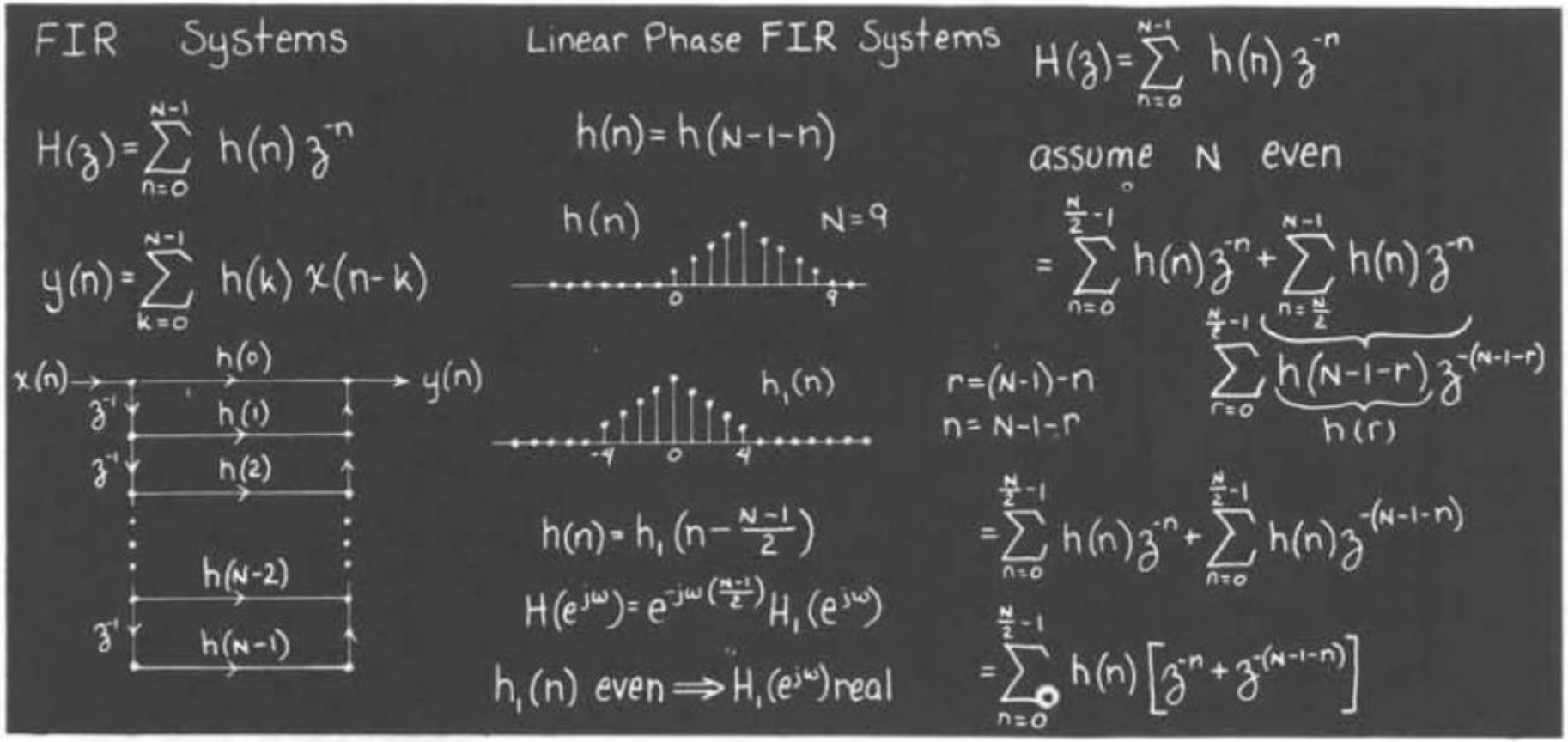
$$= \sum_{k=1}^{N_1} \frac{A_k}{1 - c_k z^{-1}} + \sum_{k=1}^{N_2} \frac{B_k(1 - e_k z^{-1})}{(1 - d_k z^{-1})(1 - d_k^* z^{-1})} + \sum_{k=0}^{M-N} c_k z^{-k}$$

$$= \sum_{k=1}^{\frac{N+1}{2}} \frac{(\gamma_{0k} + \gamma_{1k} z^{-1})}{1 - \alpha_{1k} z^{-1} - \alpha_{2k} z^{-2}} + \sum_{k=0}^{M-N} c_k z^{-k}$$

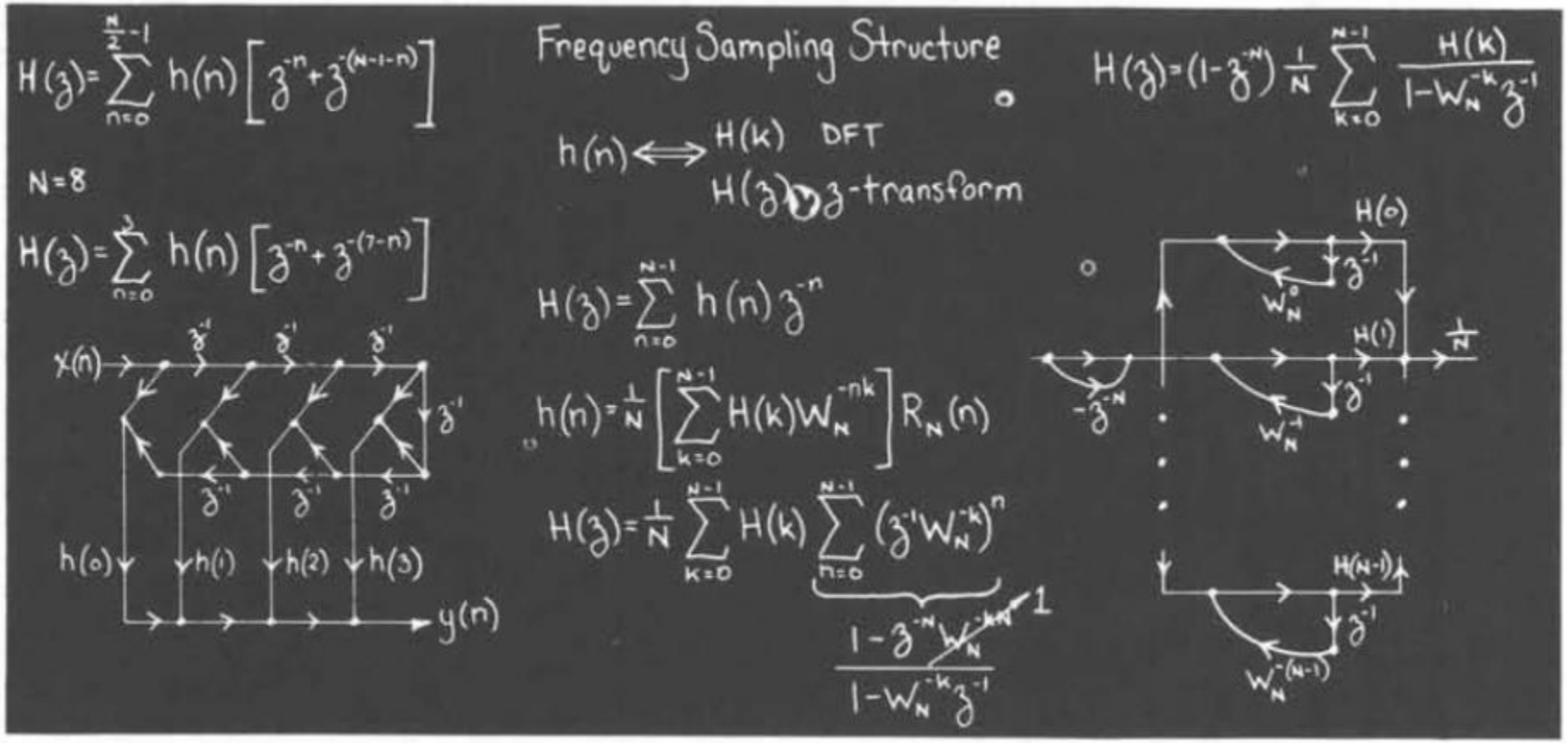


Parallel form realization with real and complex poles grouped in pairs.

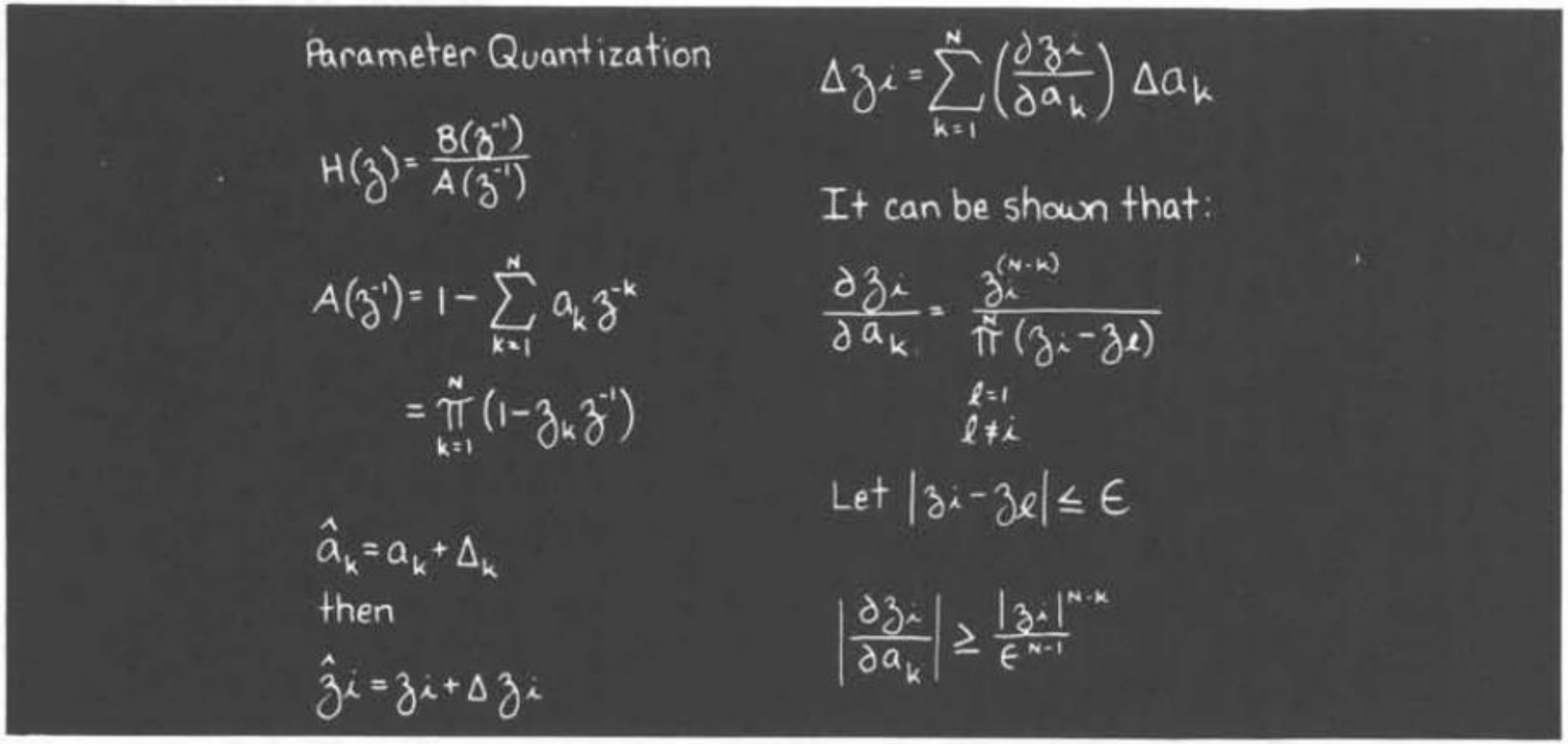
Class Review



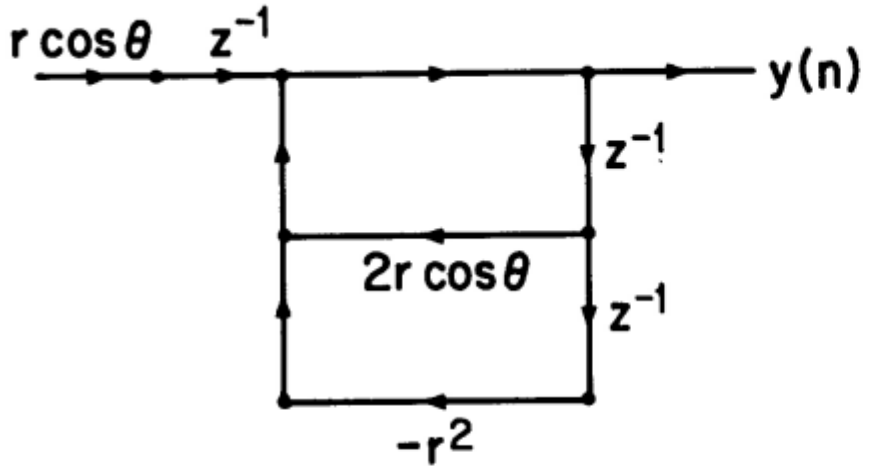
Class Review



Class Review



Class Review

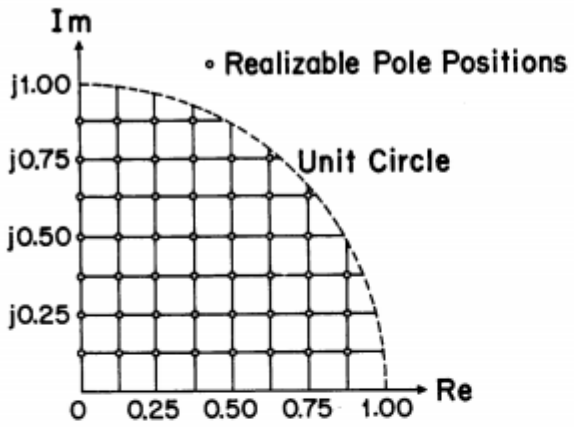


Direct-form implementation of a complex conjugate pole pair.

$$H(z) = \frac{r \cos\theta \, z^{-1}}{\left(1 - re^{j\theta} z^{-1}\right)\left(1 - re^{-j\theta} z^{-1}\right)}$$

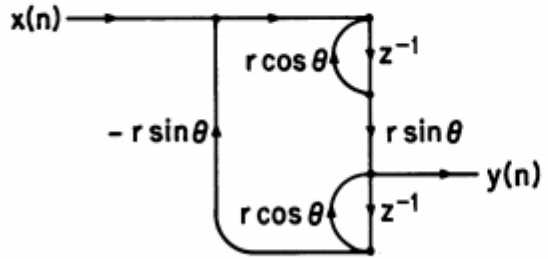
Class Review

d.



Grid of possible pole locations for the network of viewgraph d when the coefficients are quantized to three bits.

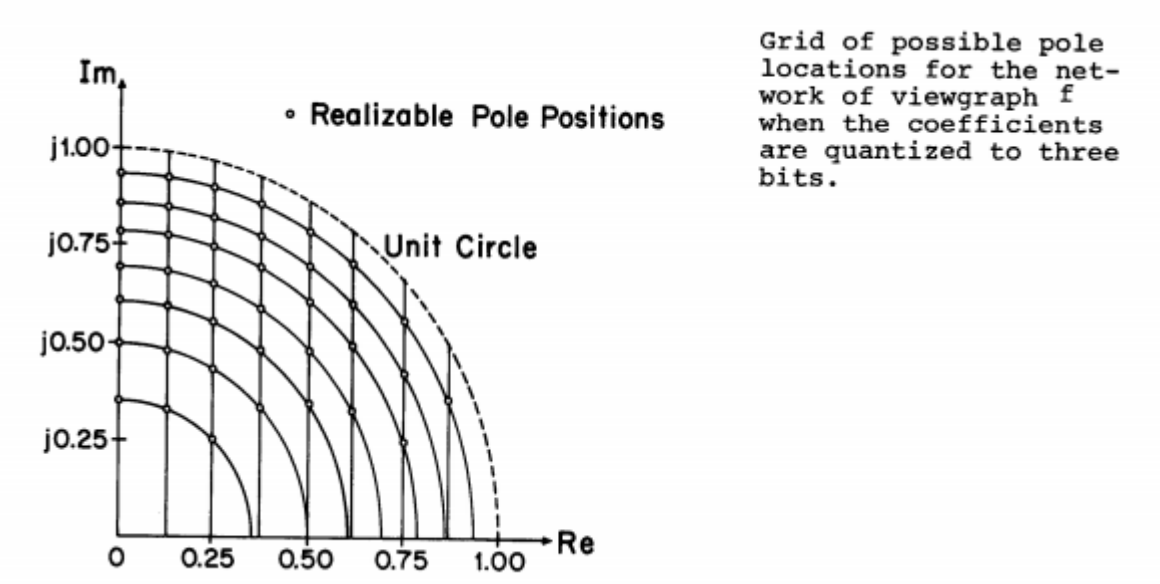
e.



$$H(z) = \frac{r \sin\theta\, z^{-1}}{(1 - re^{j\theta} z^{-1})(1 - re^{-j\theta} z^{-1})}$$

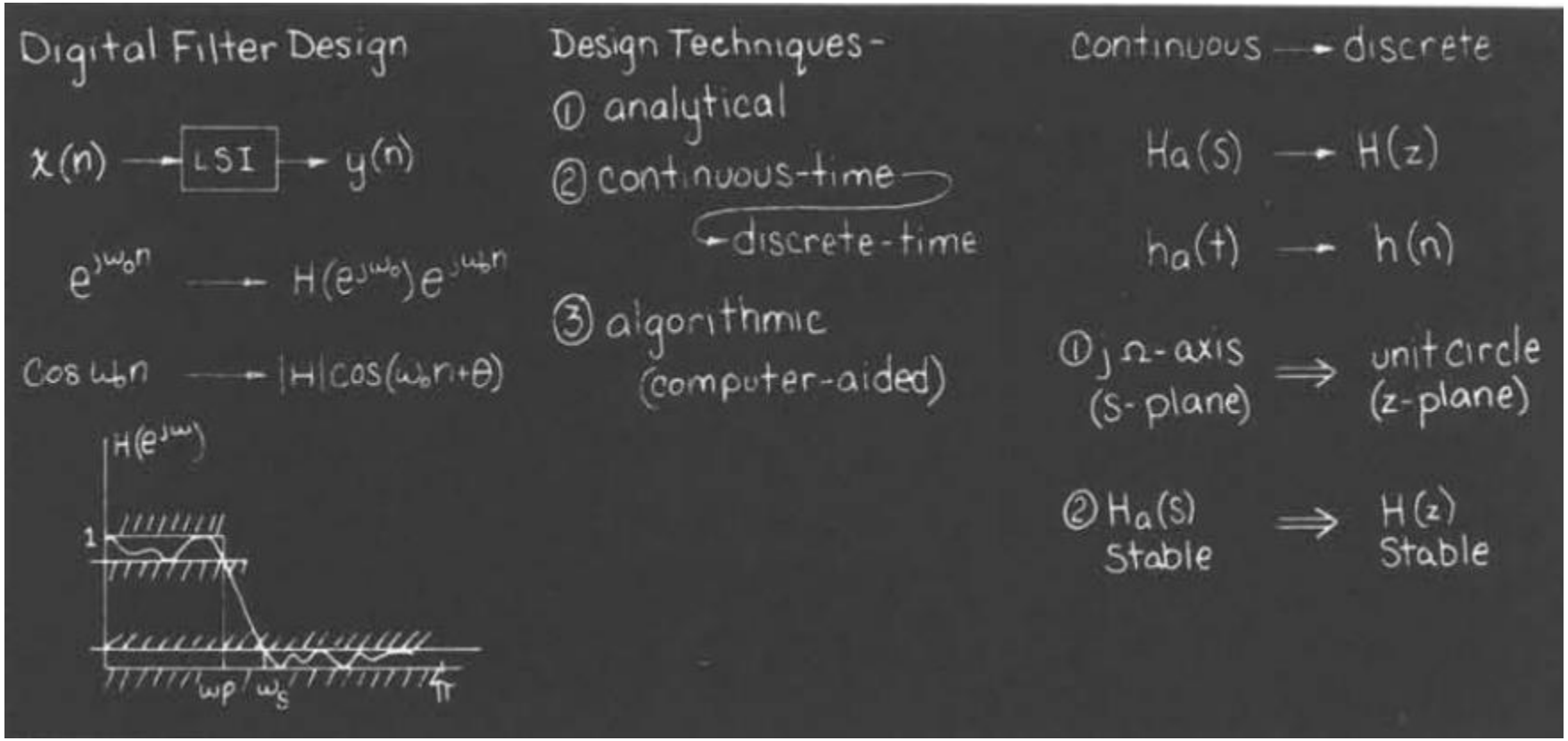
Coupled form implementation of a complex conjugate pole pair. (Note that the transfer function has been corrected. The numerator factor is rsin θ not rcos θ as indicated in the lecture.)

Class Review

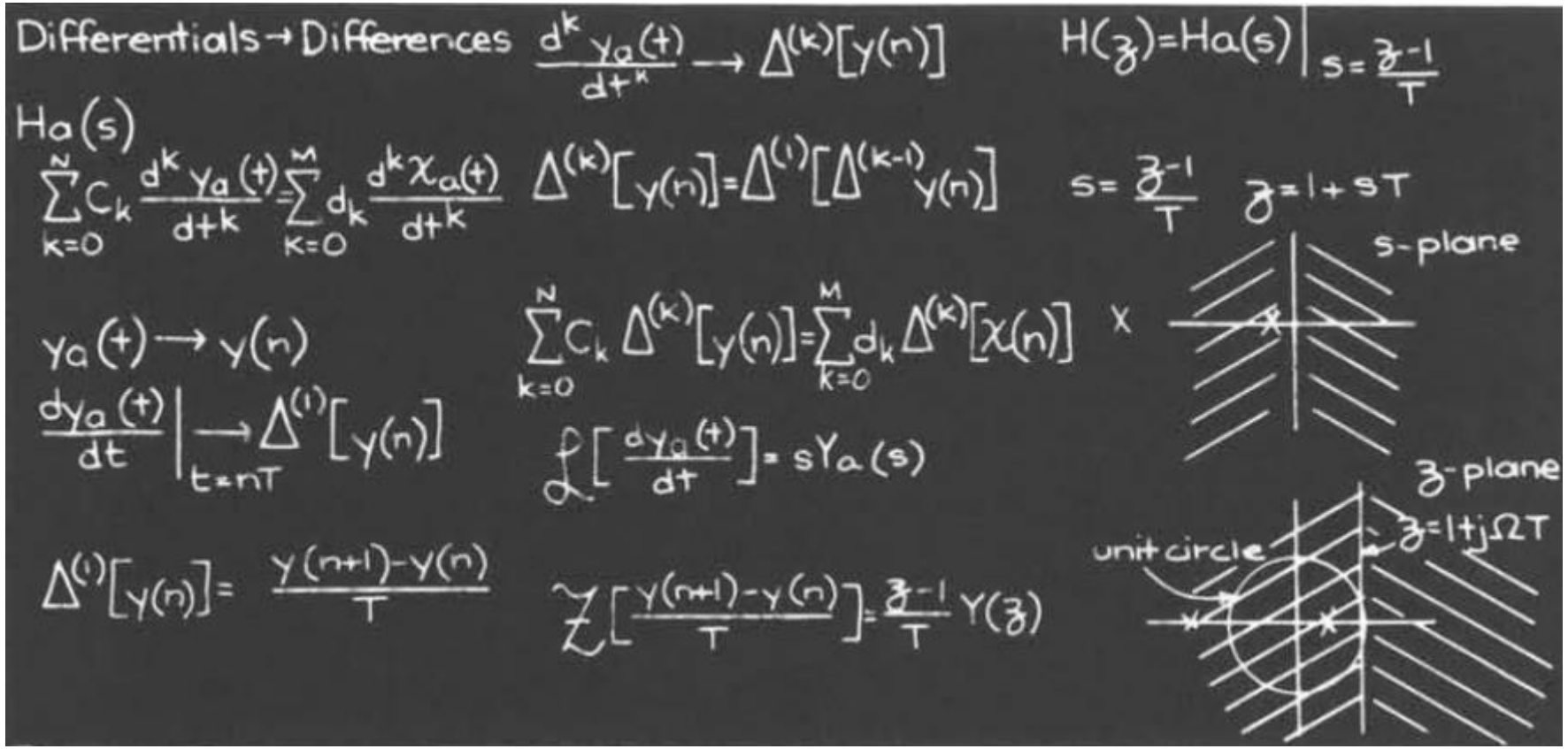


Grid of possible pole locations for the network of viewgraph f when the coefficients are quantized to three bits.

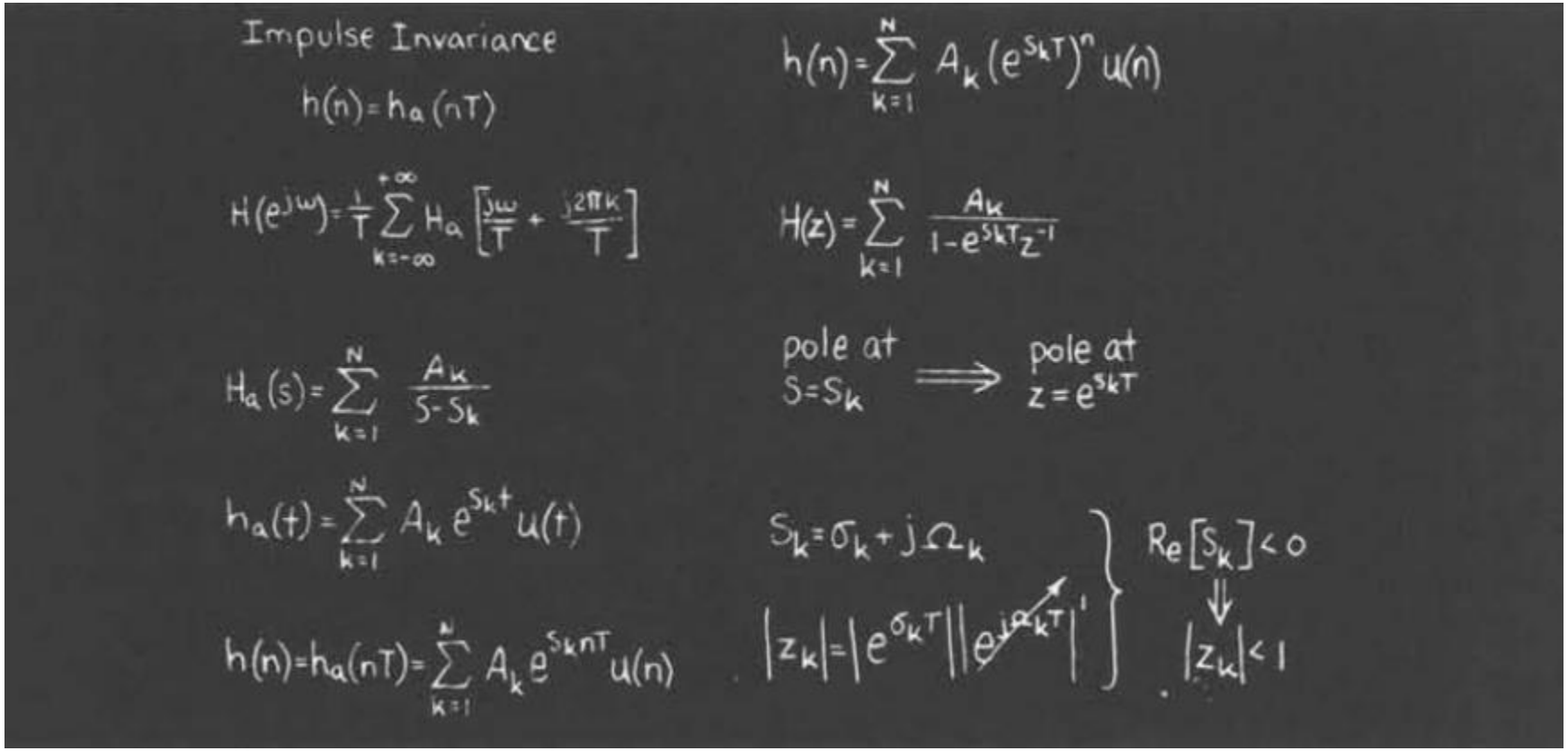
Class Review



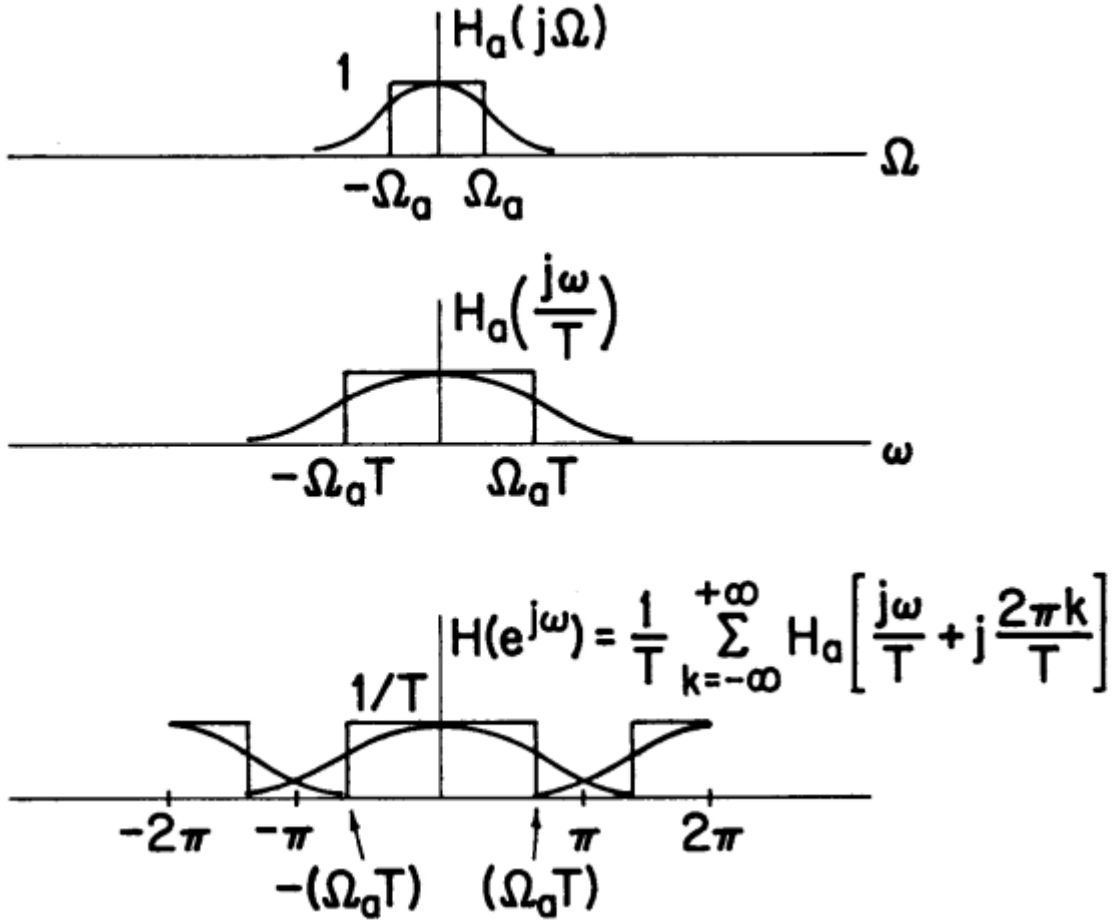
Class Review



Class Review

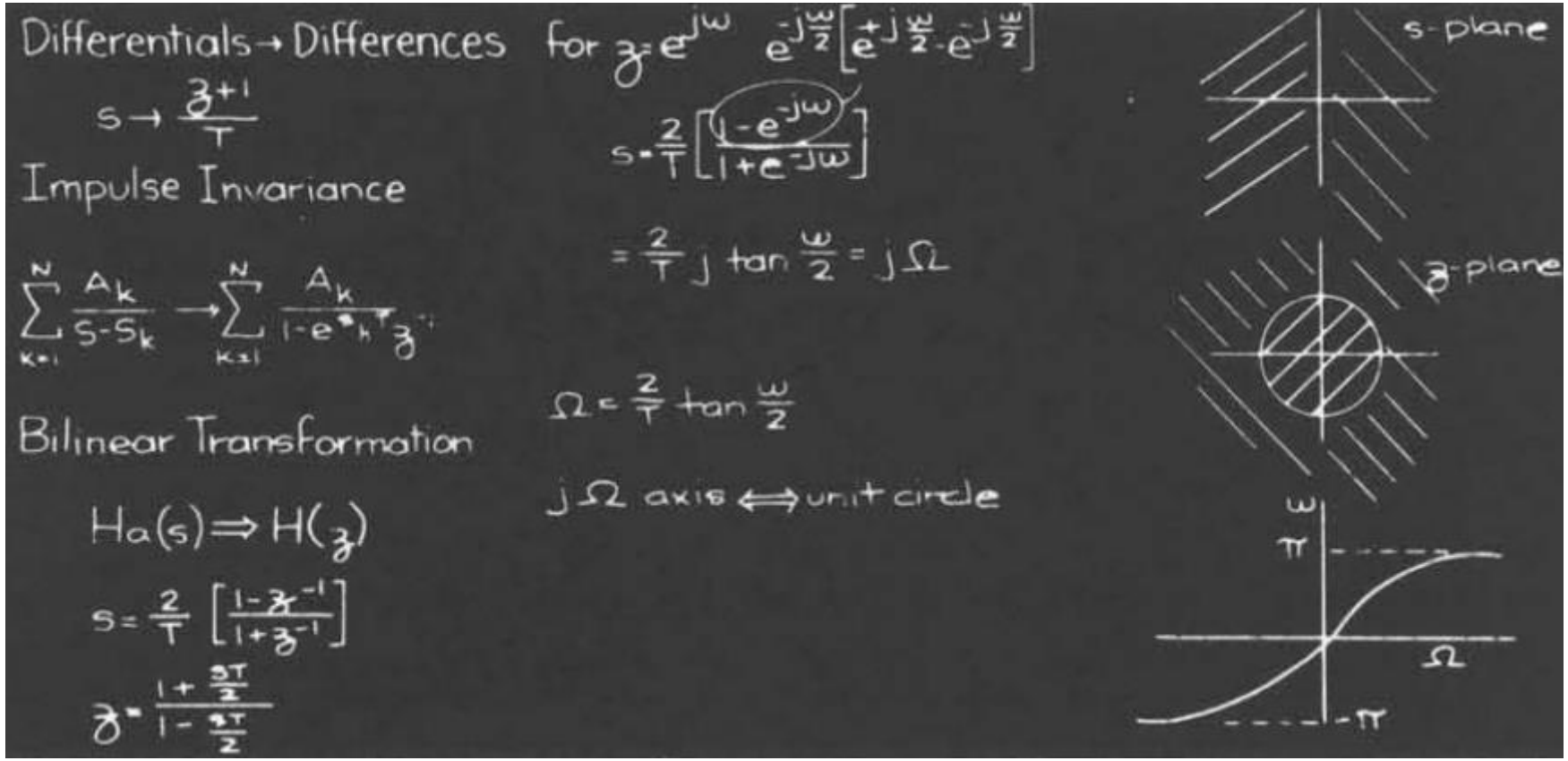


Class Review

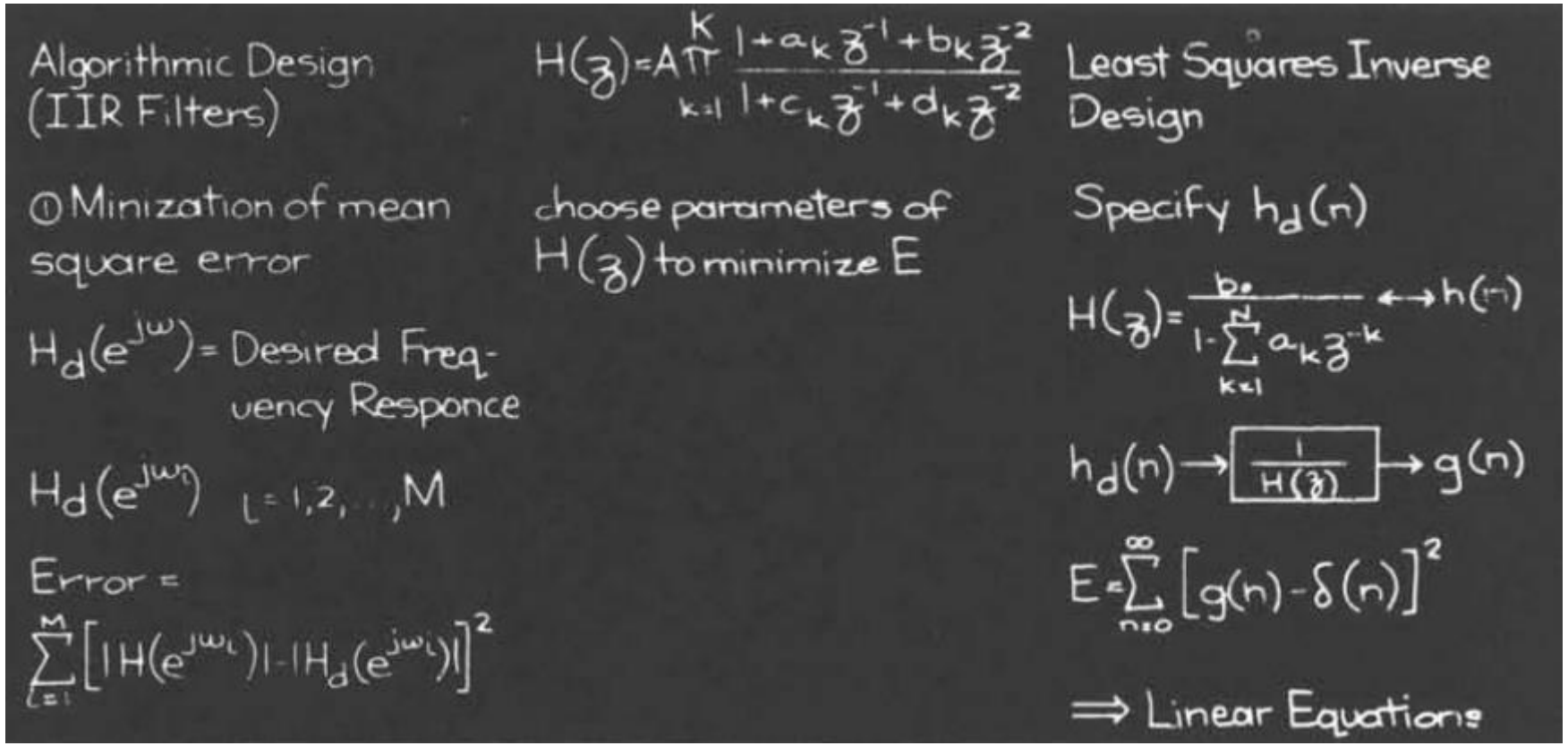


An analog frequency response and the corresponding digital frequency response obtained through impulse invariance.

Class Review



Class Review



Class Review

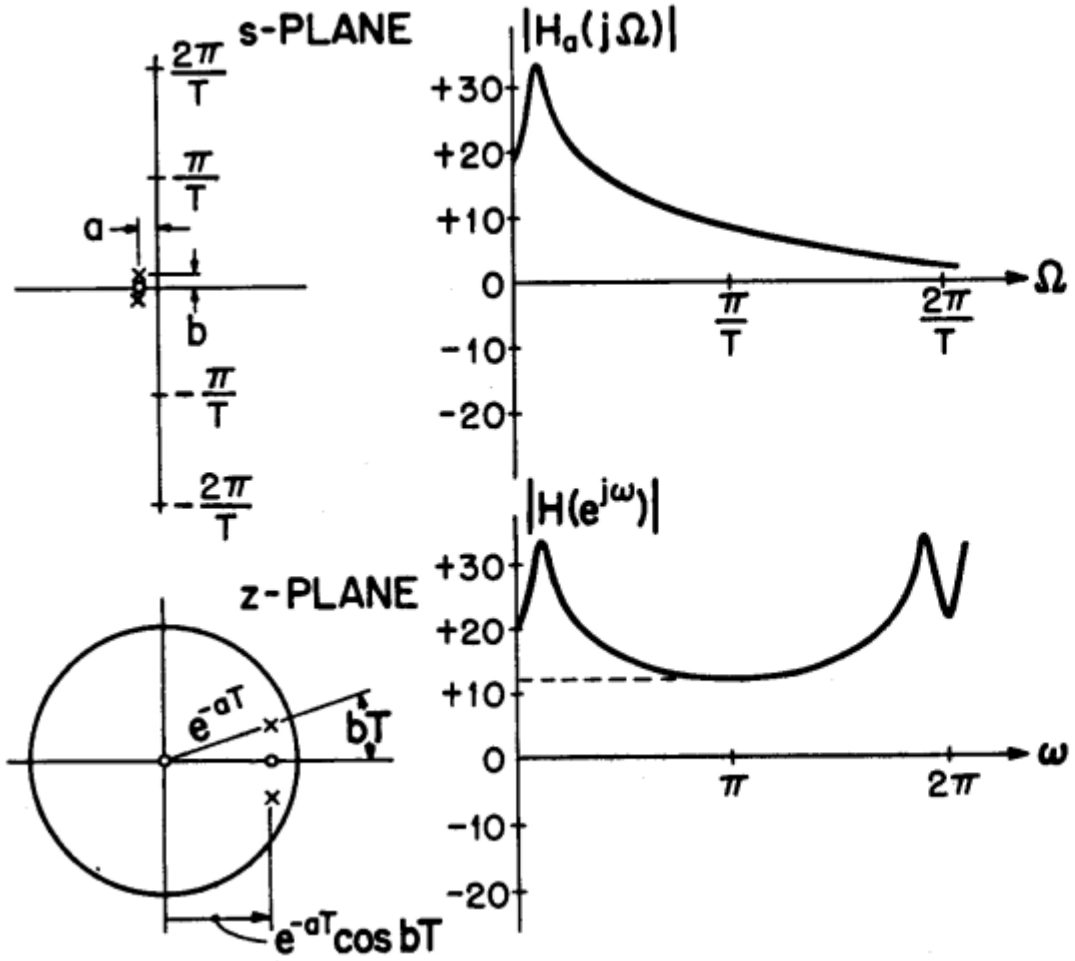
$$H_a(S) = \frac{(S+a)}{(S+a)^2 + b^2} = \frac{1/2}{S+a+jb} + \frac{1/2}{S+a-jb}$$

$$H(z) = \frac{1/2}{1-e^{-aT}e^{-jbT}z^{-1}} + \frac{1/2}{1-e^{-aT}e^{jbT}z^{-1}}$$

$$= \frac{1-(e^{-aT}\cos bT)\,z^{-1}}{(1-e^{-aT}e^{-jbT}z^{-1})(1-e^{-aT}e^{jbT}z^{-1})}$$

Example of impulse invariance.

Class Review



Pole-zero patterns and frequency response corresponding to the example of viewgraph a.

Class Review

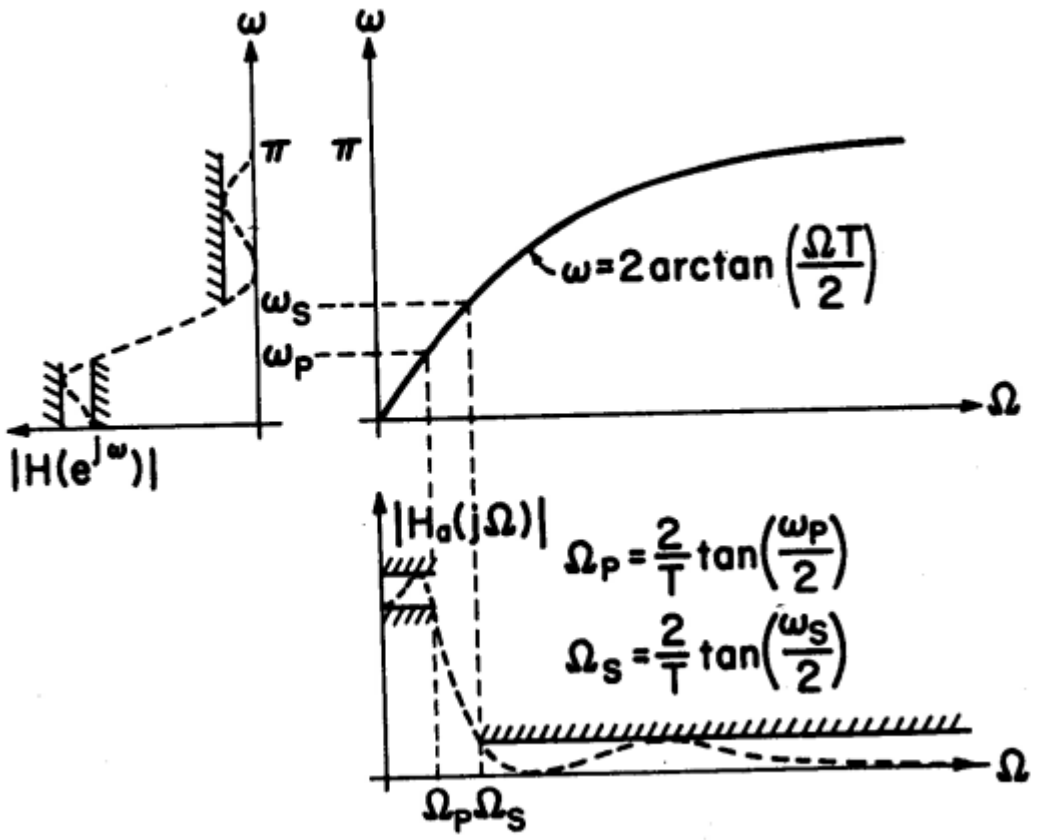


Illustration of effect of frequency warping inherent in the bi-linear transformation.

Class Review

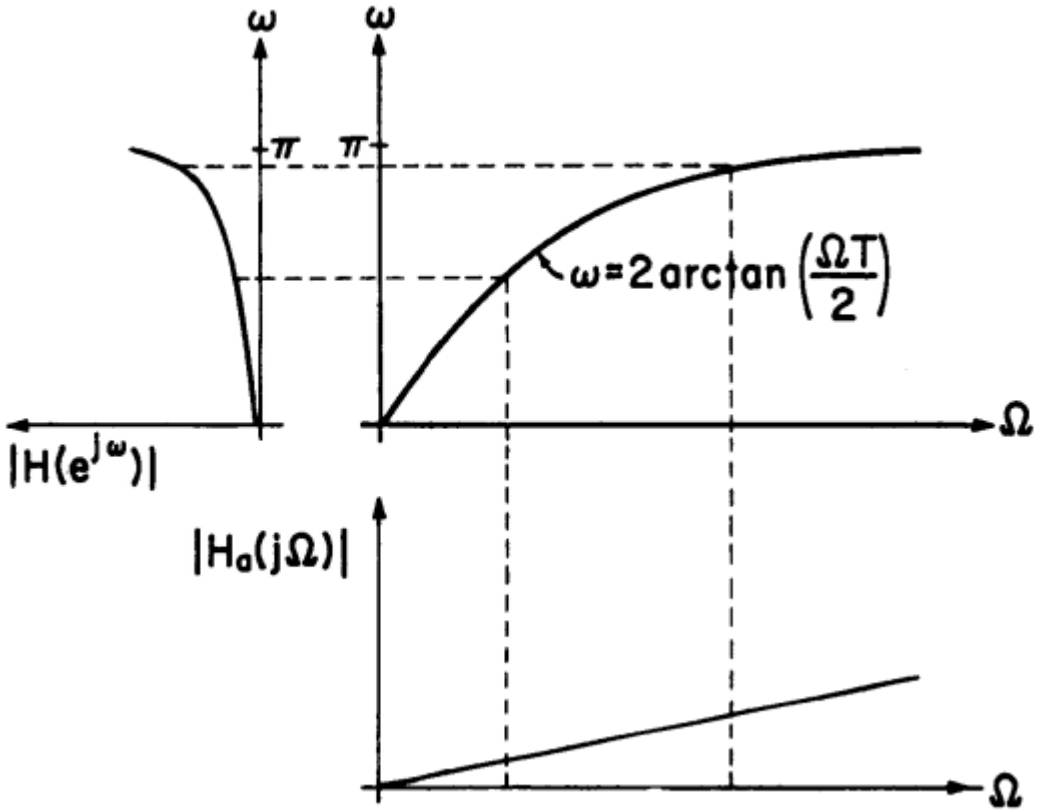


Illustration of effect of bilinear transformation on a piecewise constant frequency response characteristic.

Class Review

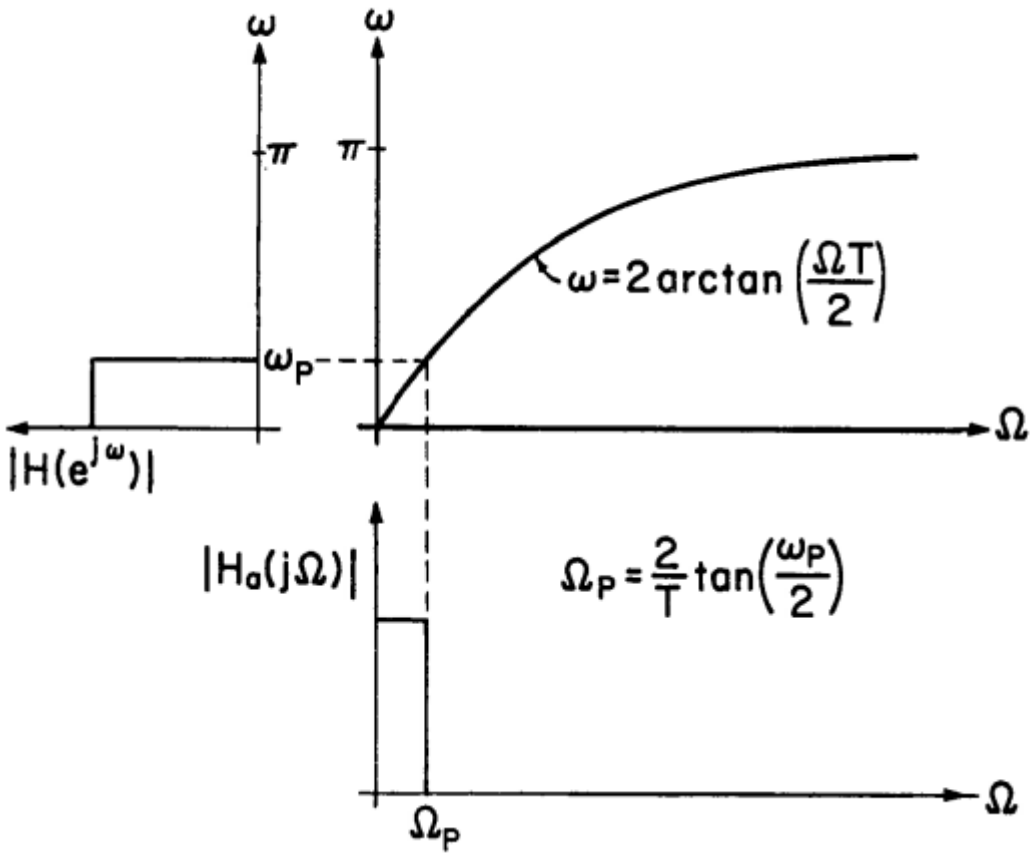
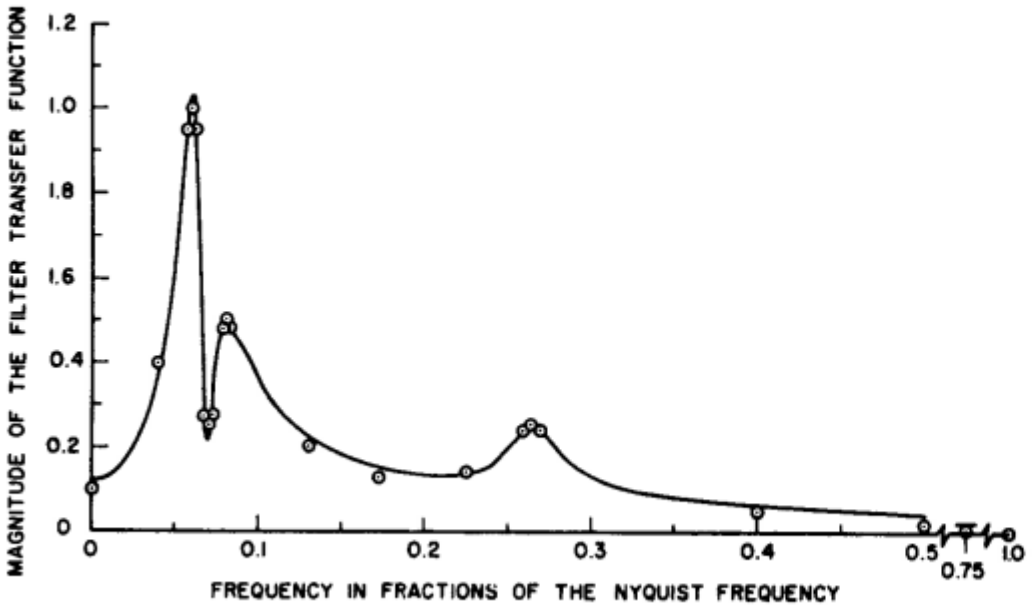


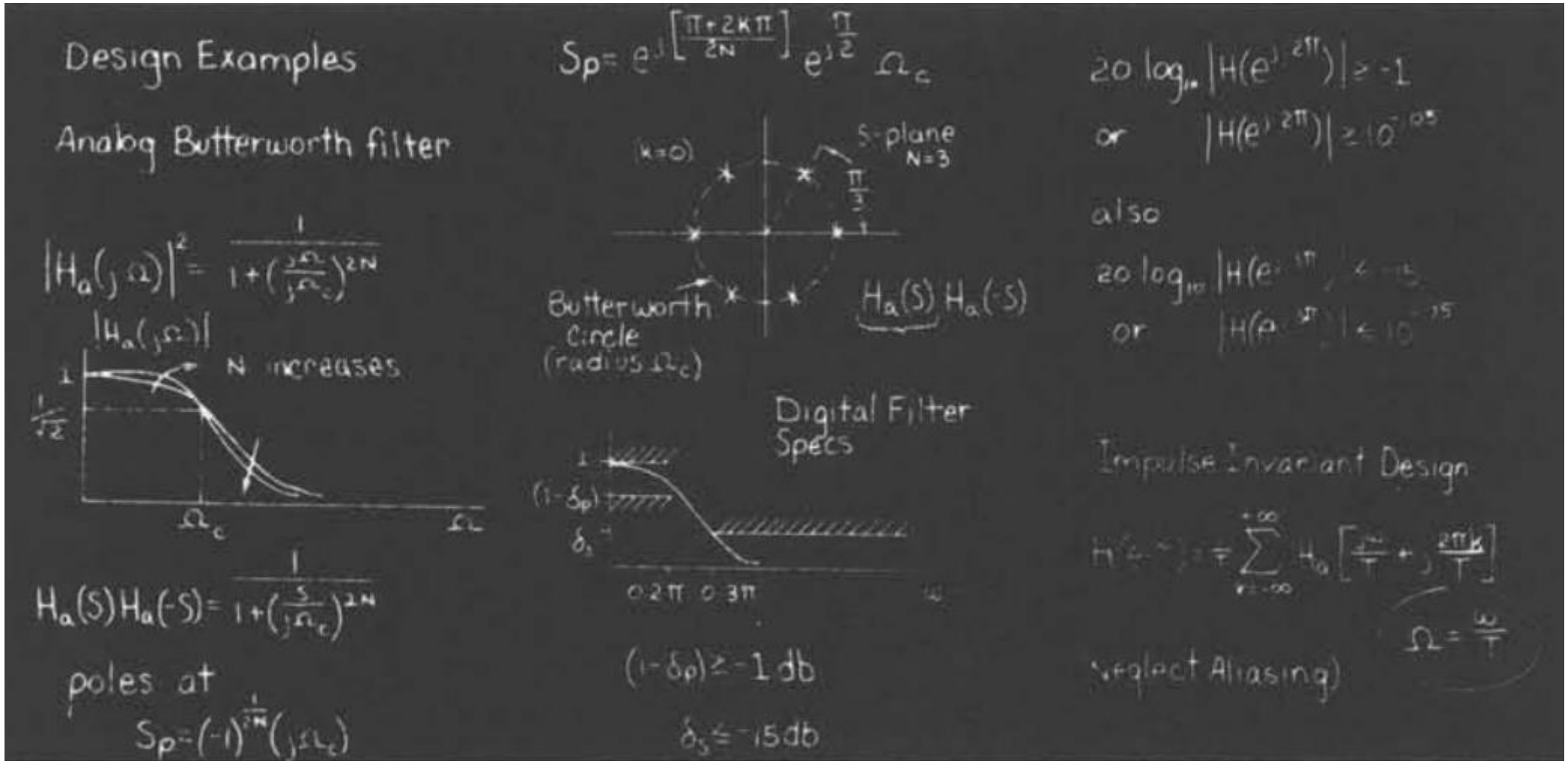
Illustration of effect of bilinear transformation on an equiripple frequency response characteristic.

Class Review

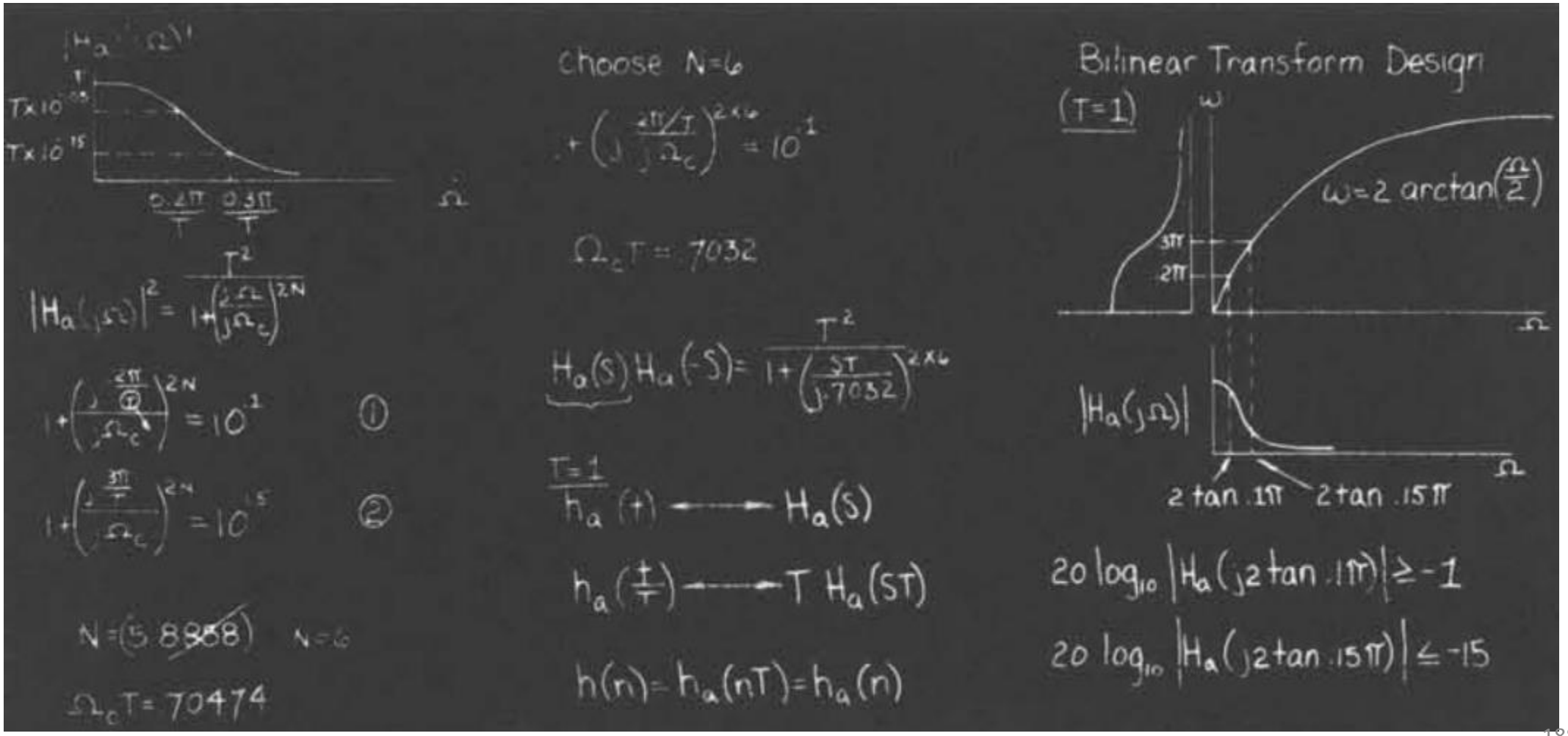


Example of frequency response obtained for an IIR filter designed by minimization of mean-square error.

Class Review



Class Review



Class Review

$$1+\left[\frac{j2\tan(.1\pi)}{j\Omega_c}\right]^{2N}=10^{1} \quad ①$$

$$1+\left[\frac{j2\tan(.15\pi)}{j\Omega_c}\right]^{2N}=10^{1.5} \quad ②$$

$$\Rightarrow N=5.30466$$

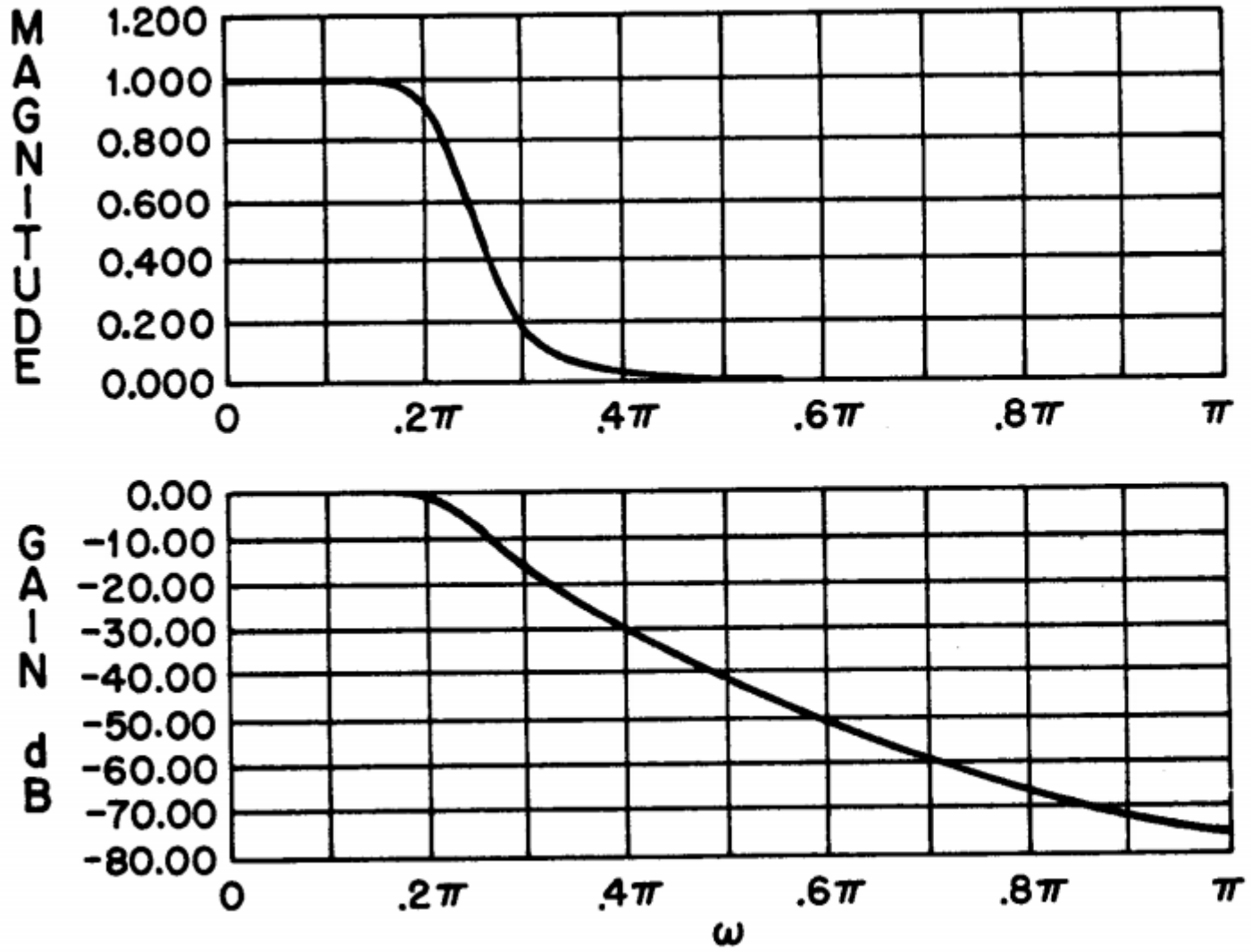
choose $N=6$ $H_a(s)$ $Ha(s)$

Meet stopband
exceed passband

$$1+\left[\frac{j2\tan(.15\pi)}{j\Omega_c}\right]^{2\times 6}=10^{1.5}$$

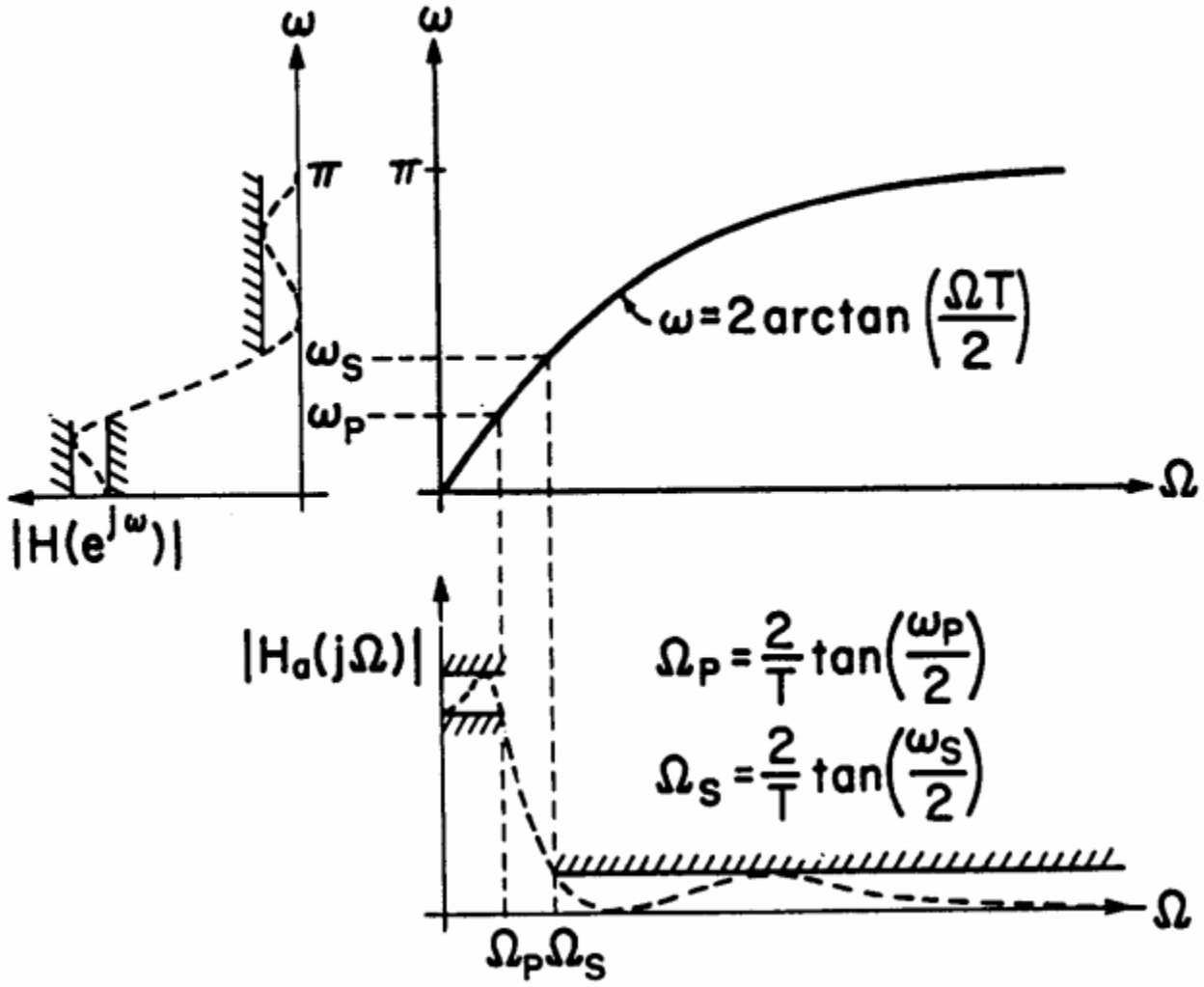
$$\Rightarrow \Omega_c=.76622$$

Class Review



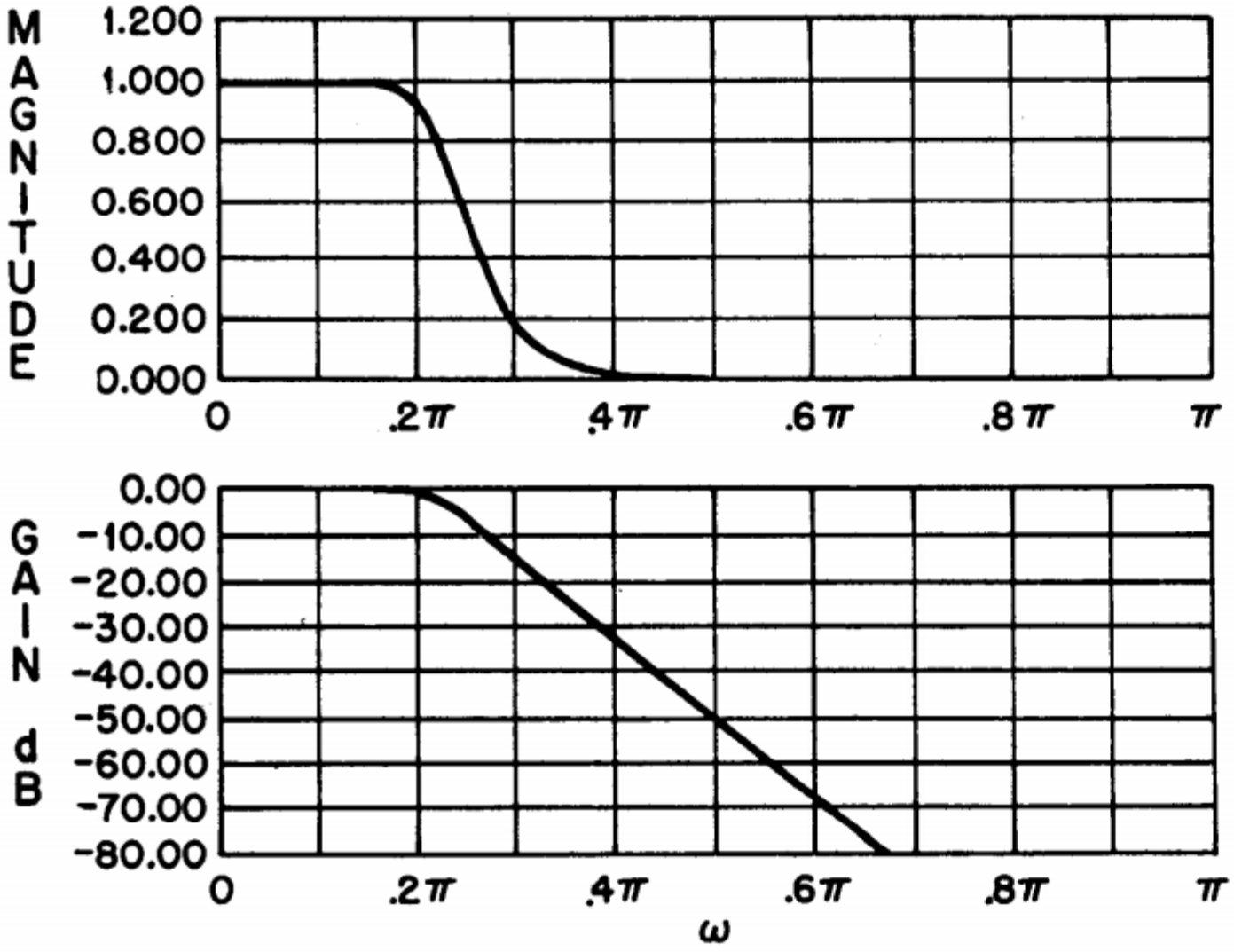
Frequency response of sixth-order digital Butterworth filter obtained by using impulse invariance.

Class Review



The bilinear transformation.

Class Review



Frequency response of sixth-order digital Butterworth filter obtained by using the bilinear transformation.

Class Review

DESIGN OF FIR DIGITAL FILTERS

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

FOR LINEAR PHASE $h(n) = h(N-1-n)$

BASIC DESIGN METHODS:

① WINDOWS

② FREQUENCY SAMPLING

③ EQUIRIPPLE DESIGN

Basic design methods for FIR digital filters

a.

DESIGN OF FIR FILTERS USING WINDOWS

DESIRED UNIT SAMPLE RESPONSE: $h_d(n)$

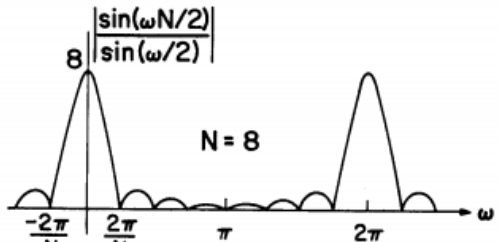
$$h(n) = w(n)\, h_d(n)$$

$$w(n) = 0 \qquad n < 0,\ n > (N-1)$$

$$H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\theta})\, W[e^{j(\omega-\theta)}]\, d\theta$$

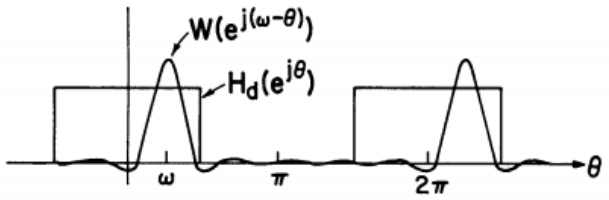
Design of FIR filters using the window method.

b.

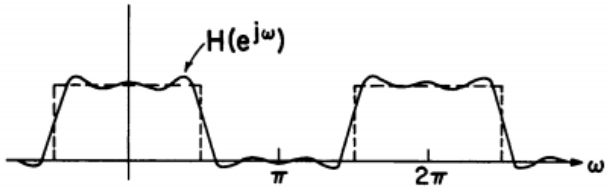


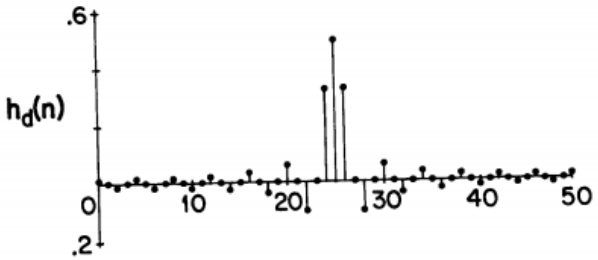
Magnitude of the Fourier transform for an eight point rectangular window.

Class Review

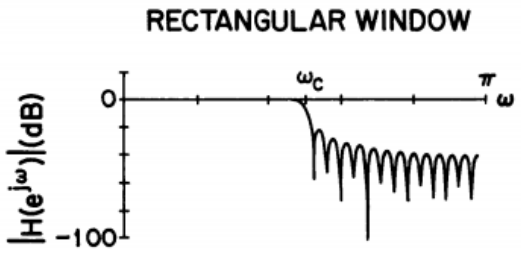


Effect of convolving the Fourier transform of a rectangular window with an ideal low pass filter characteristic.



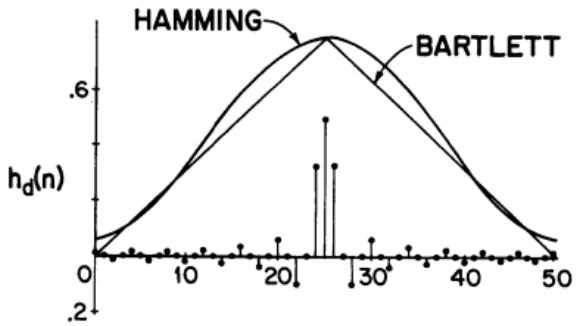


Unit-sample response of an ideal low-pass filter truncated by a 51-point rectangular window.

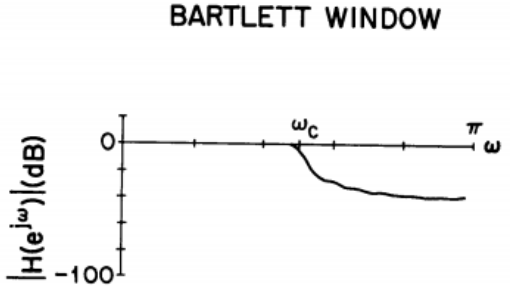


Frequency response corresponding to the unit-sample response in viewgraph e.

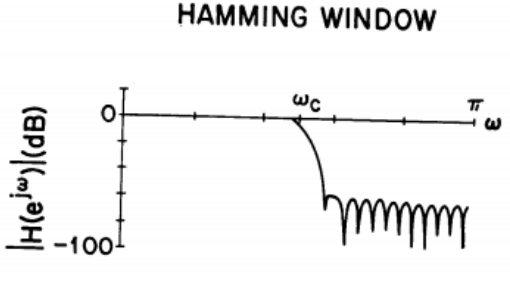
Class Review



The Hamming and Bartlett windows.

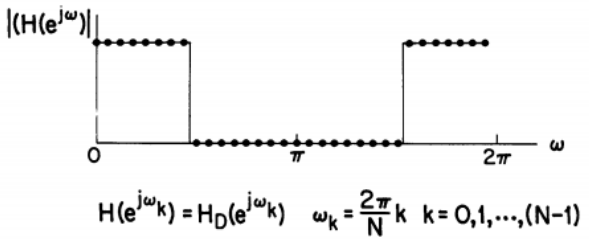


Frequency response of an FIR lowpass filter obtained by multiplying the unit-sample response of an ideal low pass filter by a Bartlett window.

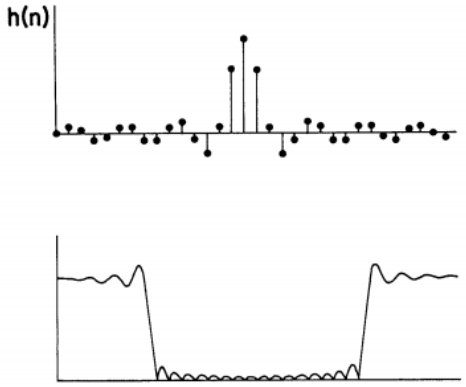


Frequency response of an FIR lowpass filter filter obtained by multiplying the unit sample response of an ideal lowpass filter by a Hamming window. (Note that the stopband attenuation is approximately 65 db not 30 db as stated in the lecture.)

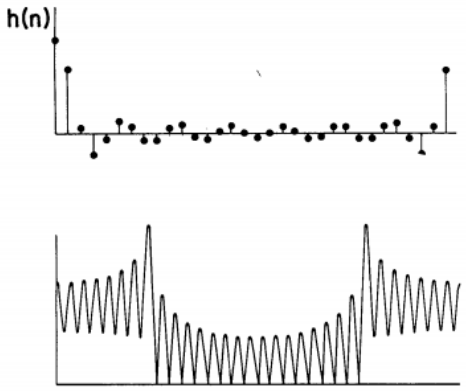
Class Review



Equally spaced frequency samples of an ideal low pass filter.

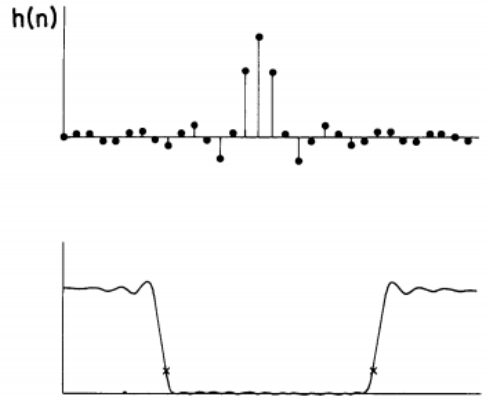


A unit-sample response the magnitude of whose DFT is equal to the frequency samples in viewgraph j.. The bottom trace is the magnitude of the Fourier transform of this unit-sample response.

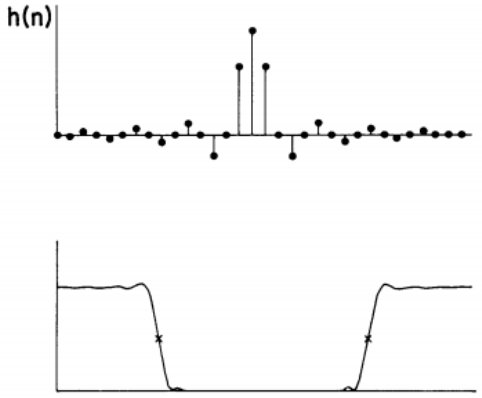


Another unit-sample response the magnitude of whose DFT is equal to the frequency samples in viewgraph j.. The bottom trace is the magnitude of the Fourier transform of this unit-sample response.

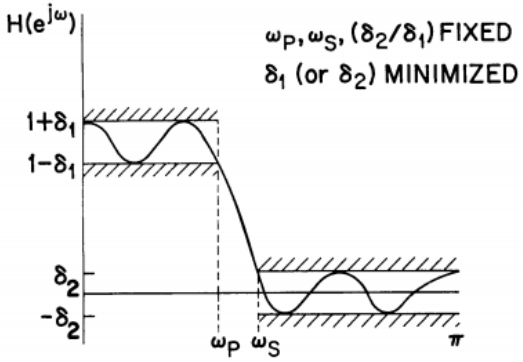
Class Review



Unit sample response and frequency respons when one frequency sample is moved from the stopband into the transition band.

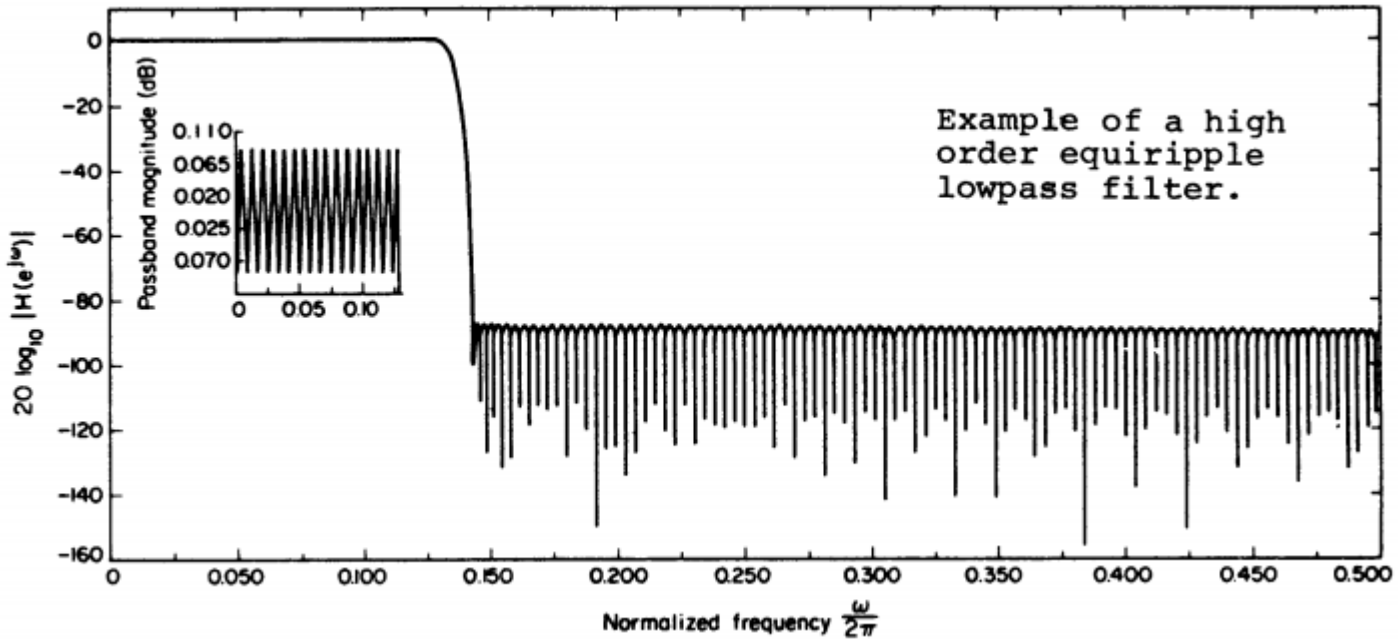


Similar to viewgraph m. with a different value of the frequenc sample in the transition band.

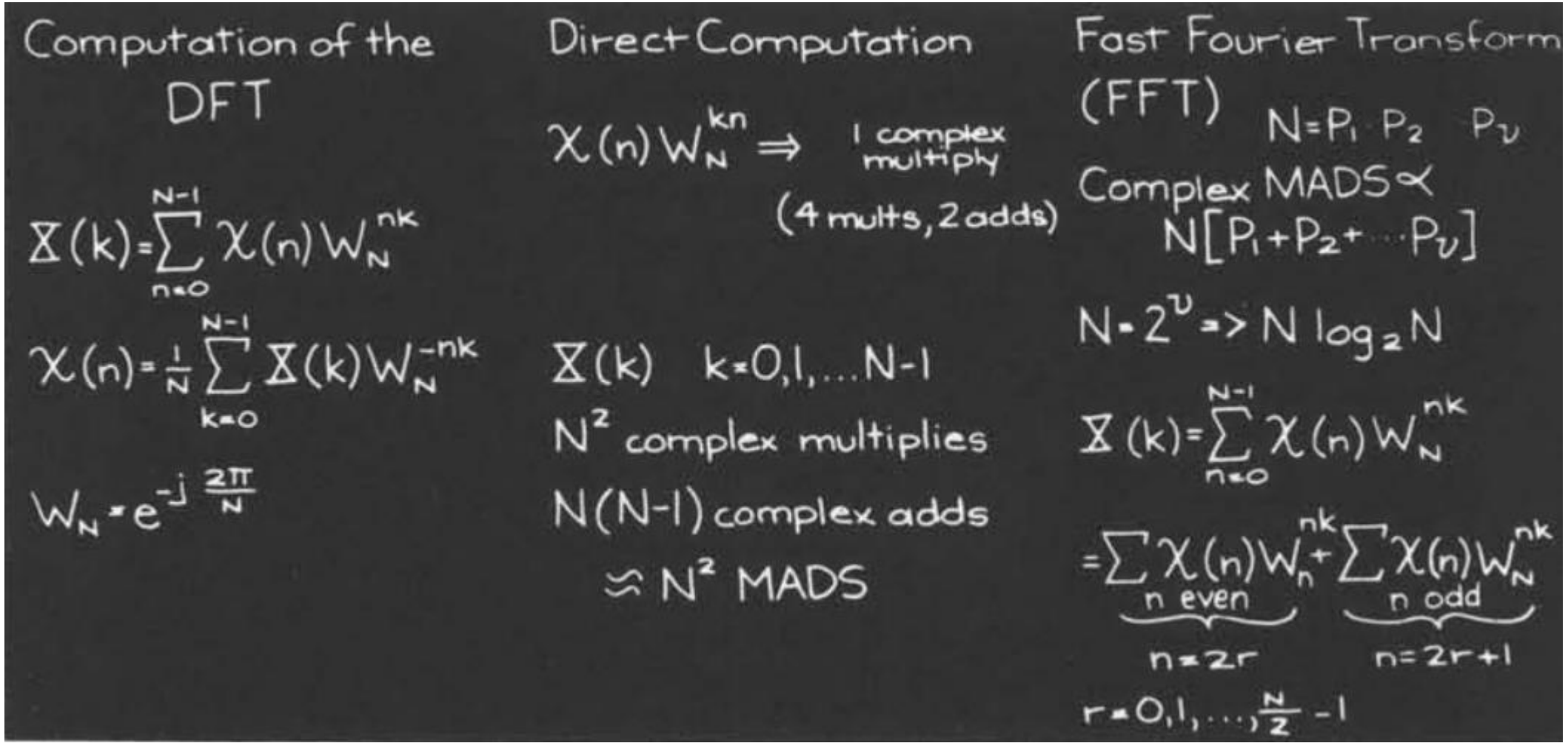


Equiripple approxi-mation of a lowpass filter.

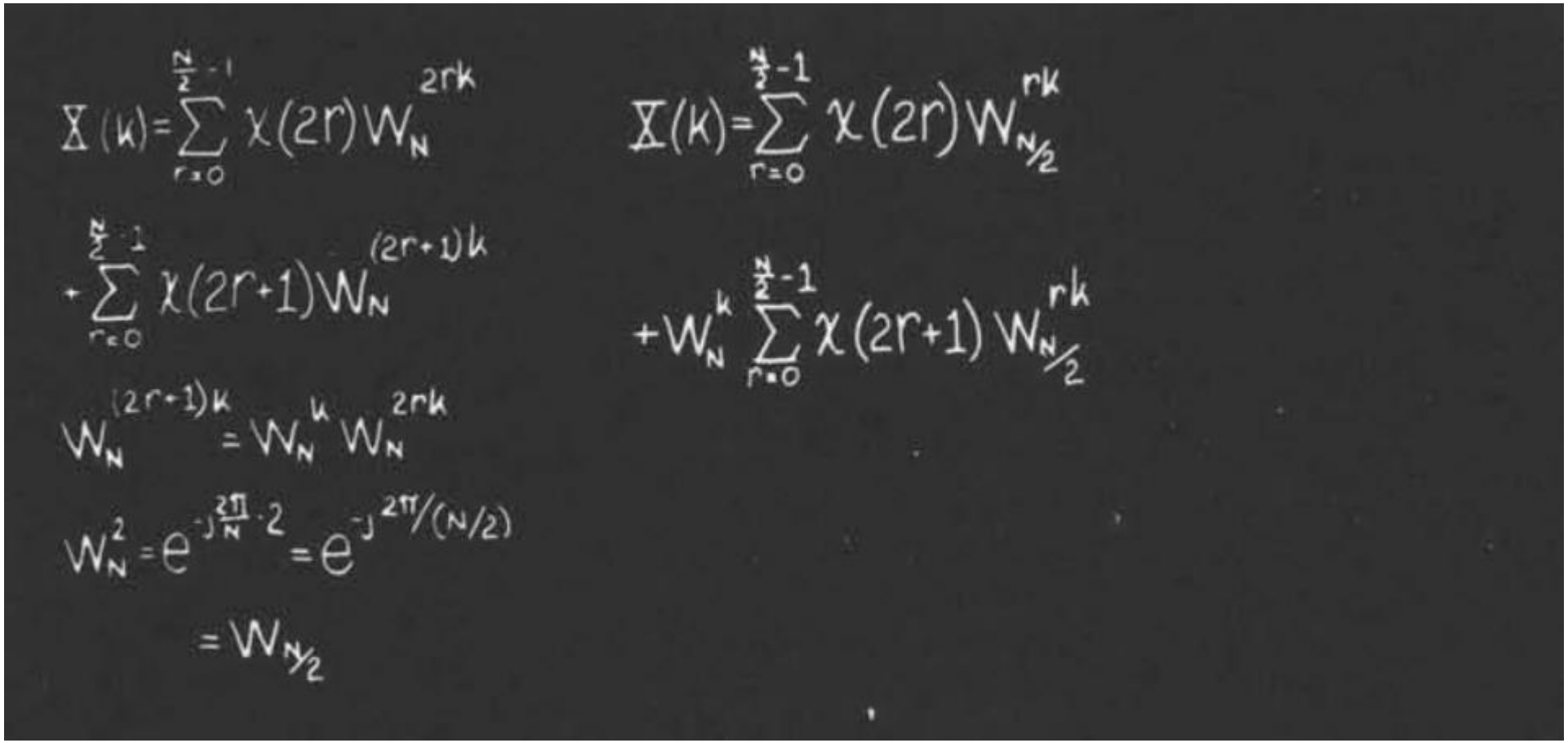
Class Review



Class Review



Class Review



Class Review

$$X(k) = \underbrace{\sum_{r=0}^{\frac{N}{2}-1} X(2r) W_{N/2}^{rk}}_{\frac{N}{2}\text{ POINT DFT}} + W_N^k \underbrace{\sum_{r=0}^{\frac{N}{2}-1} X(2r+1)\, W_{N/2}^{rk}}_{\frac{N}{2}\text{ POINT DFT}}$$

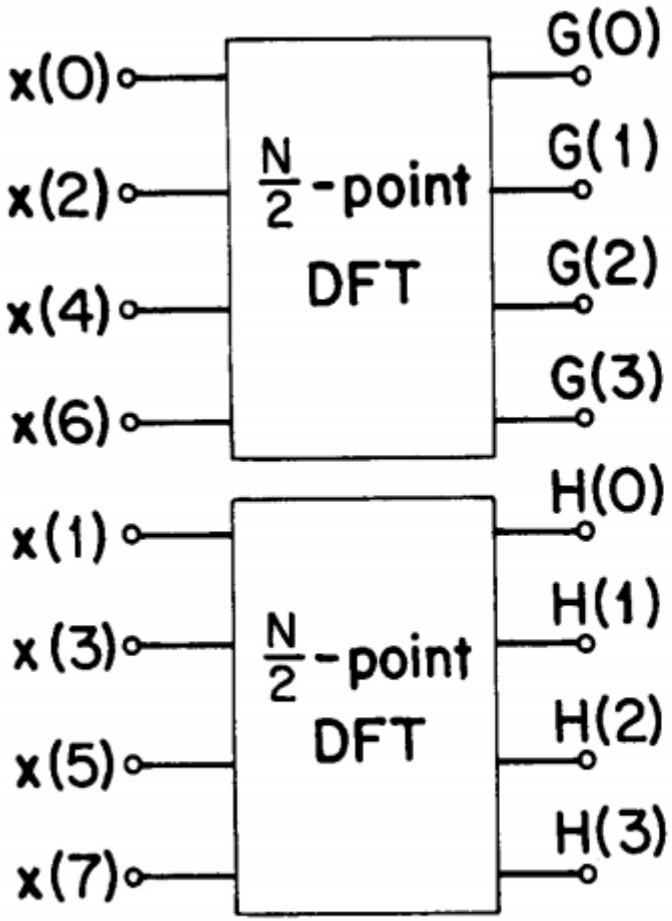
$G(k)$ $\qquad$ $H(k)$

DFT of a sequence in terms of the DFT of the even and odd numbered points.

$$X(k) = G(k) + W_N^k H(k)$$

$$2\left(\frac{N}{2}\right)^2 + N = N + \frac{N^2}{2}$$

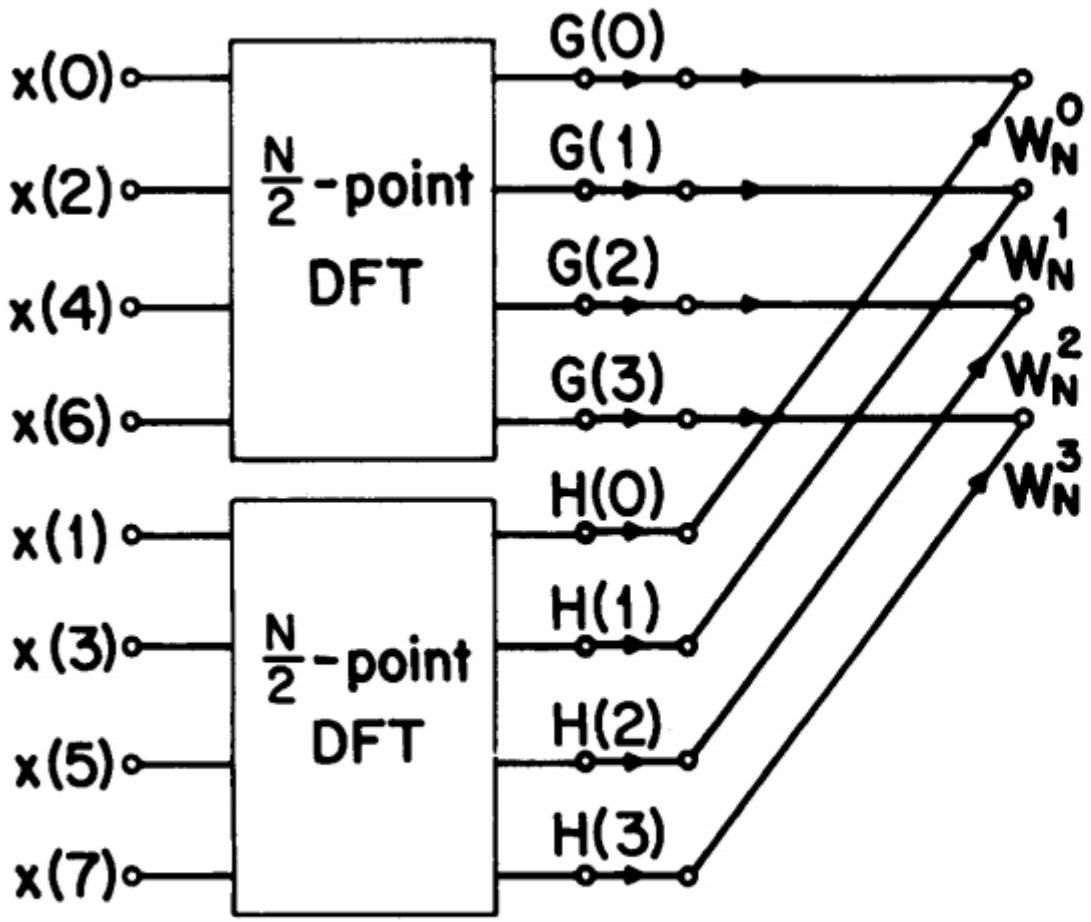
Class Review



$$X(k) = G(k) + W_N^k H(k)$$

N/2-point DFT's of even and odd-numbered points

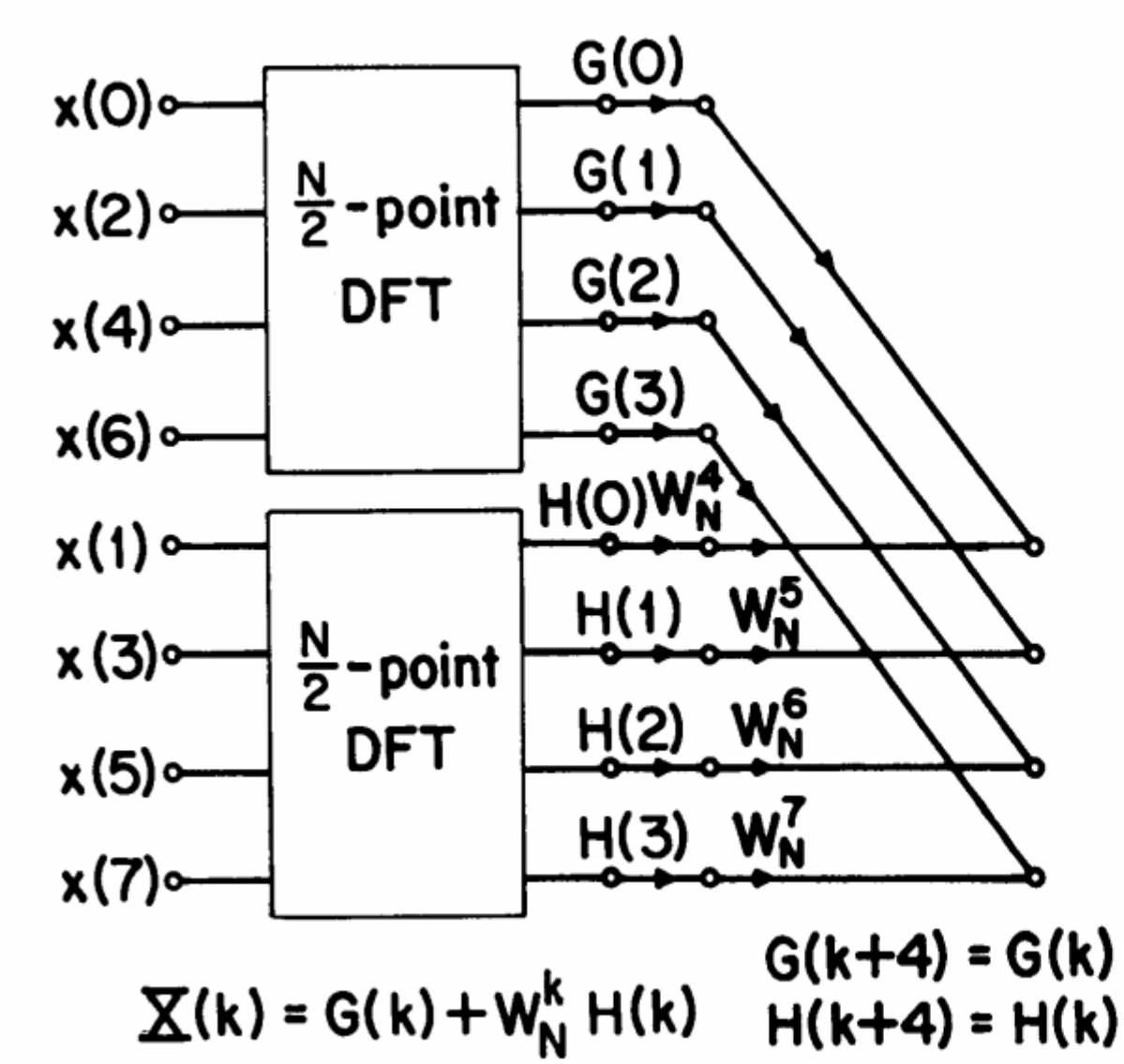
Class Review



$$X(k) = G(k) + W_N^k H(k)$$

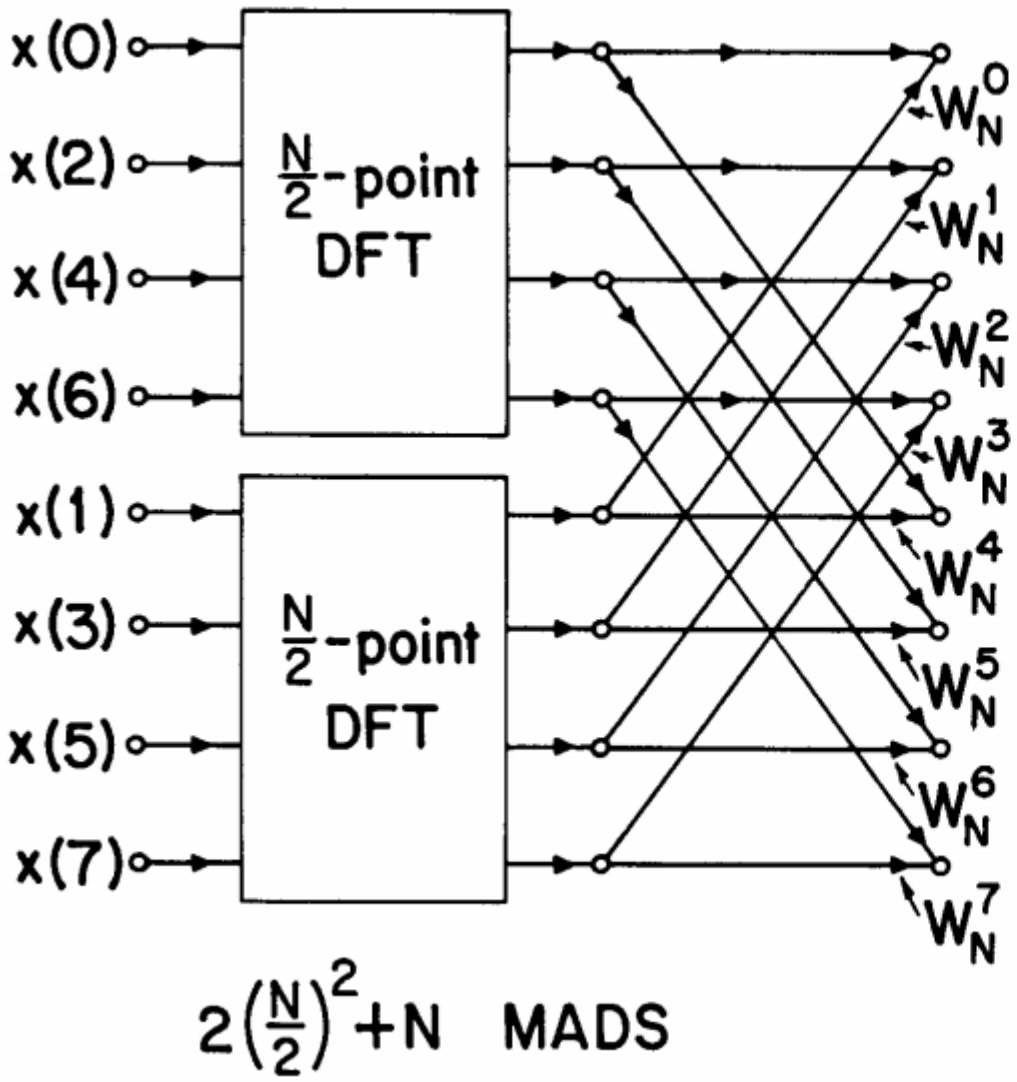
Combination of G(k) and H(k) to obtain first half of X(k)

Class Review



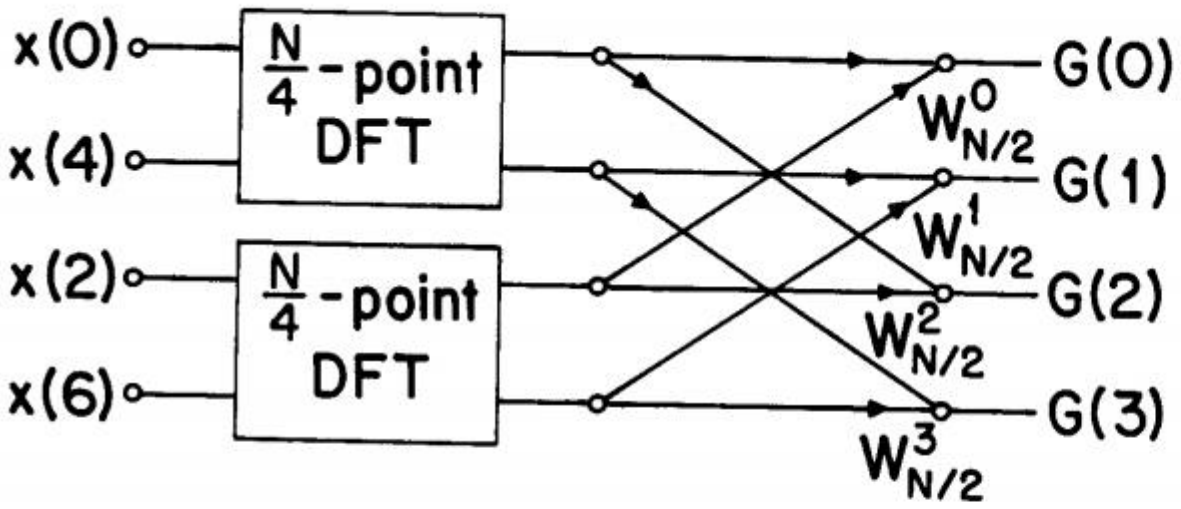
Combination of G(k) and H(k) to obtain second half of X(k)

Class Review



Combination of G(k) and H(k) to obtain X(k)

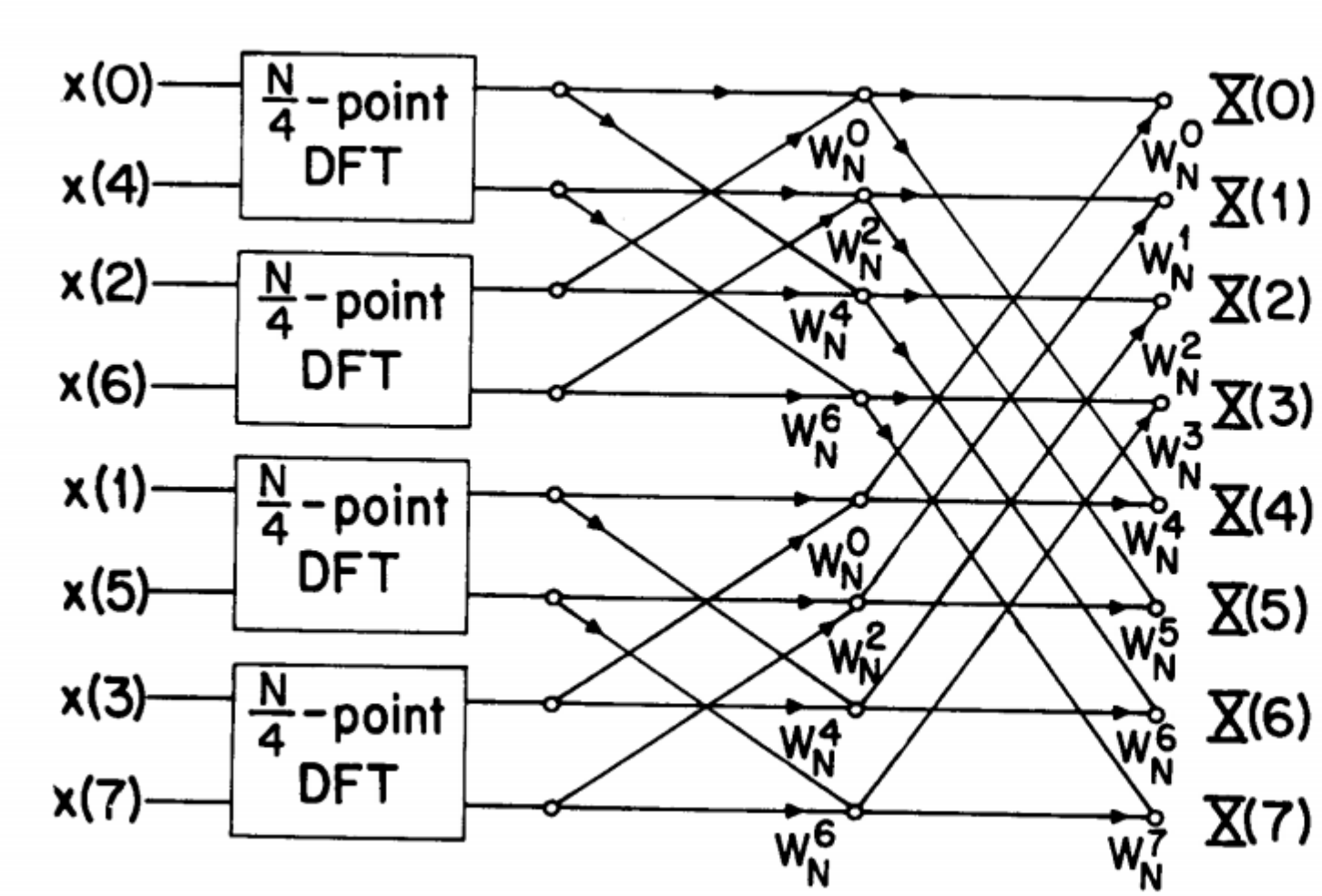
Class Review



Computation of G(k) in terms of two N/4-point DFT's

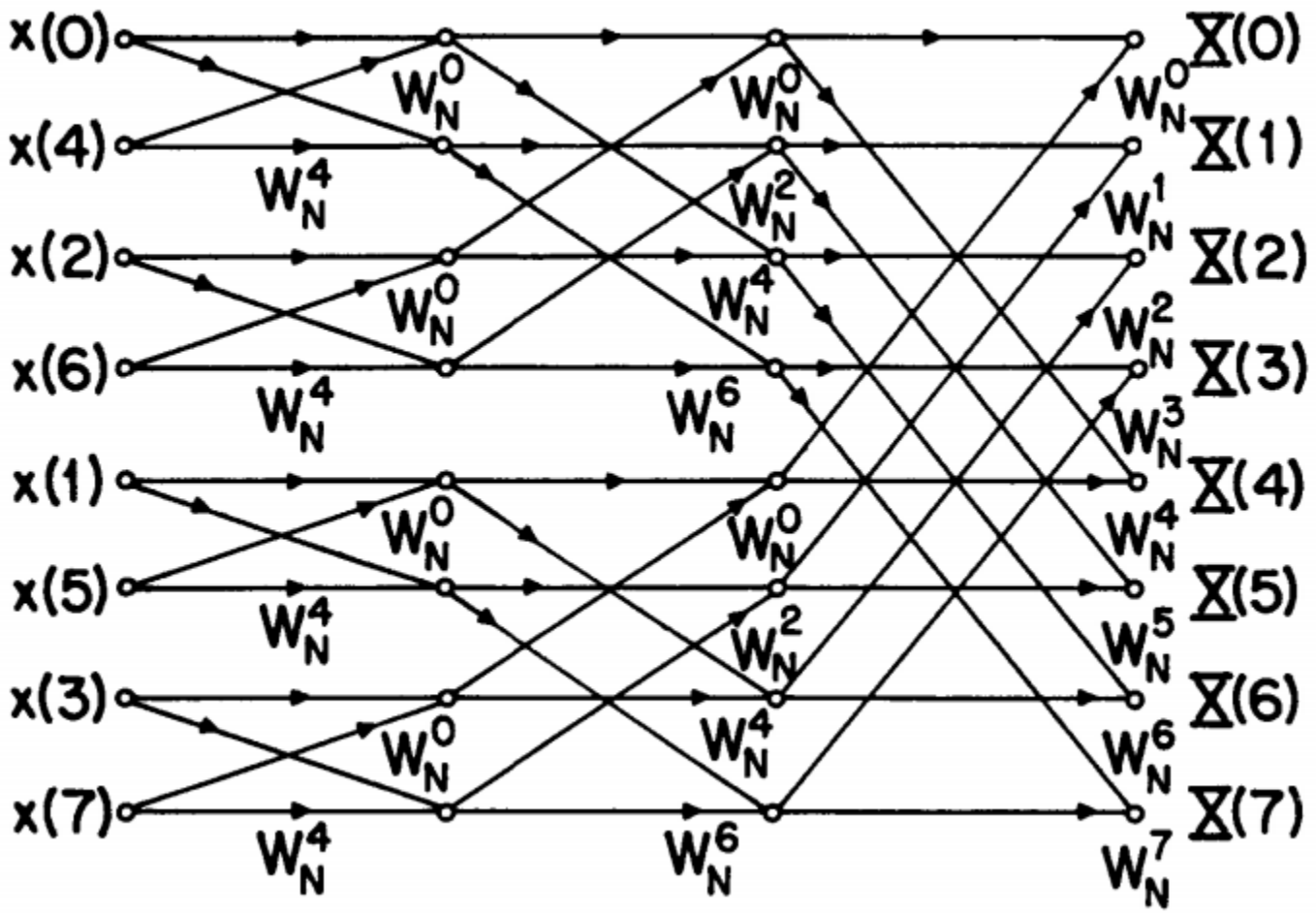
$$W_{\frac{N}{2}} = W_N^2$$

Class Review



Computation of X(k) by combining flow-graph g and h.

Class Review



Flow-graph for computation of X(k)

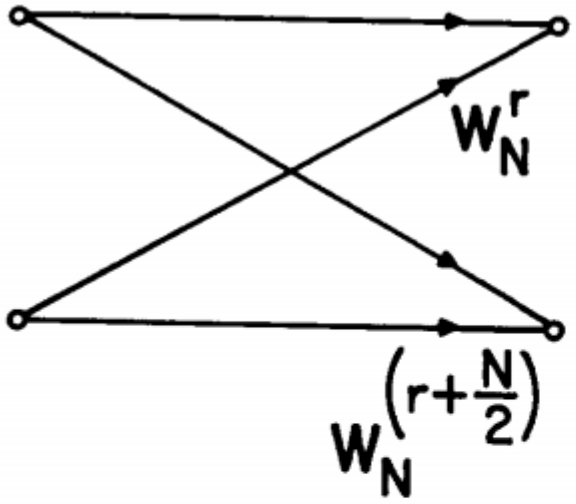
Class Review

$$N = 2^{\nu}$$

Savings in computation resulting from successive decimation in time

1 N POINT DFT	N^2	
$2\ \frac{N}{2}$ POINT DFT'S +	$2\left(\frac{N}{2}\right)^2+N$	$\left(\frac{N}{2}\right)^2 \rightarrow 2\left(\frac{N}{4}\right)^2+\frac{N}{2}$
$4\ \frac{N}{4}$ POINT DFT'S +	$4\left(\frac{N}{4}\right)^2+N+N$	$\left(\frac{N}{4}\right)^2 \rightarrow 2\left(\frac{N}{8}\right)^2+\frac{N}{4}$
$8\ \frac{N}{8}$ POINT DFT'S +	$8\left(\frac{N}{8}\right)^2+N+N+N$	$\left(\frac{N}{8}\right)^2 \rightarrow 2\left(\frac{N}{16}\right)^2+\frac{N}{8}$
	$\vdots$	
	$\underbrace{N+N+\cdots+N}_{\nu \text{ TIMES}}$	

Class Review

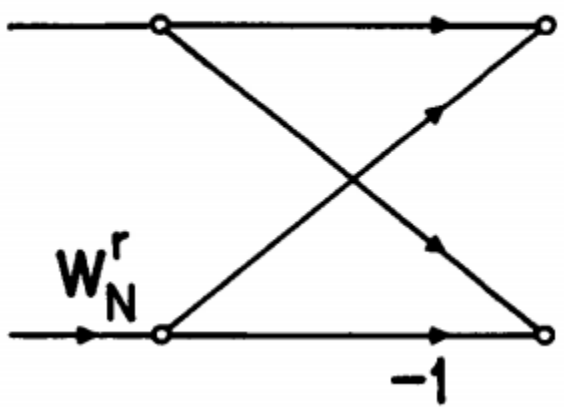


Basic butterfly computation in flow-graph j.

$$W_N^{\left(r+\frac{N}{2}\right)} = W_N^r \, W_N^{\frac{N}{2}}$$

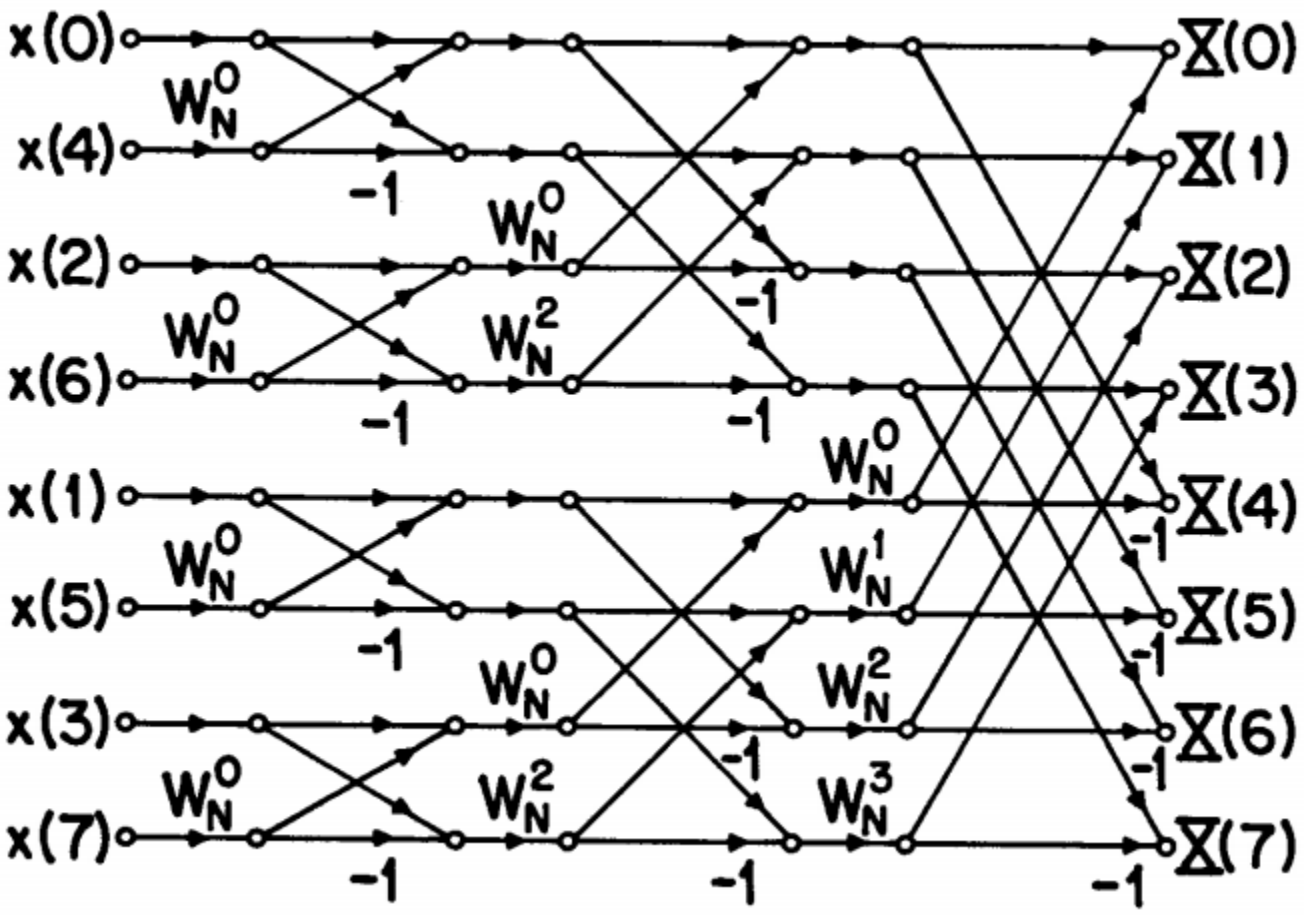
$$W_N^{\frac{N}{2}} = e^{-j\frac{2\pi}{N}\frac{N}{2}} = e^{-j\pi} = -1$$

Class Review



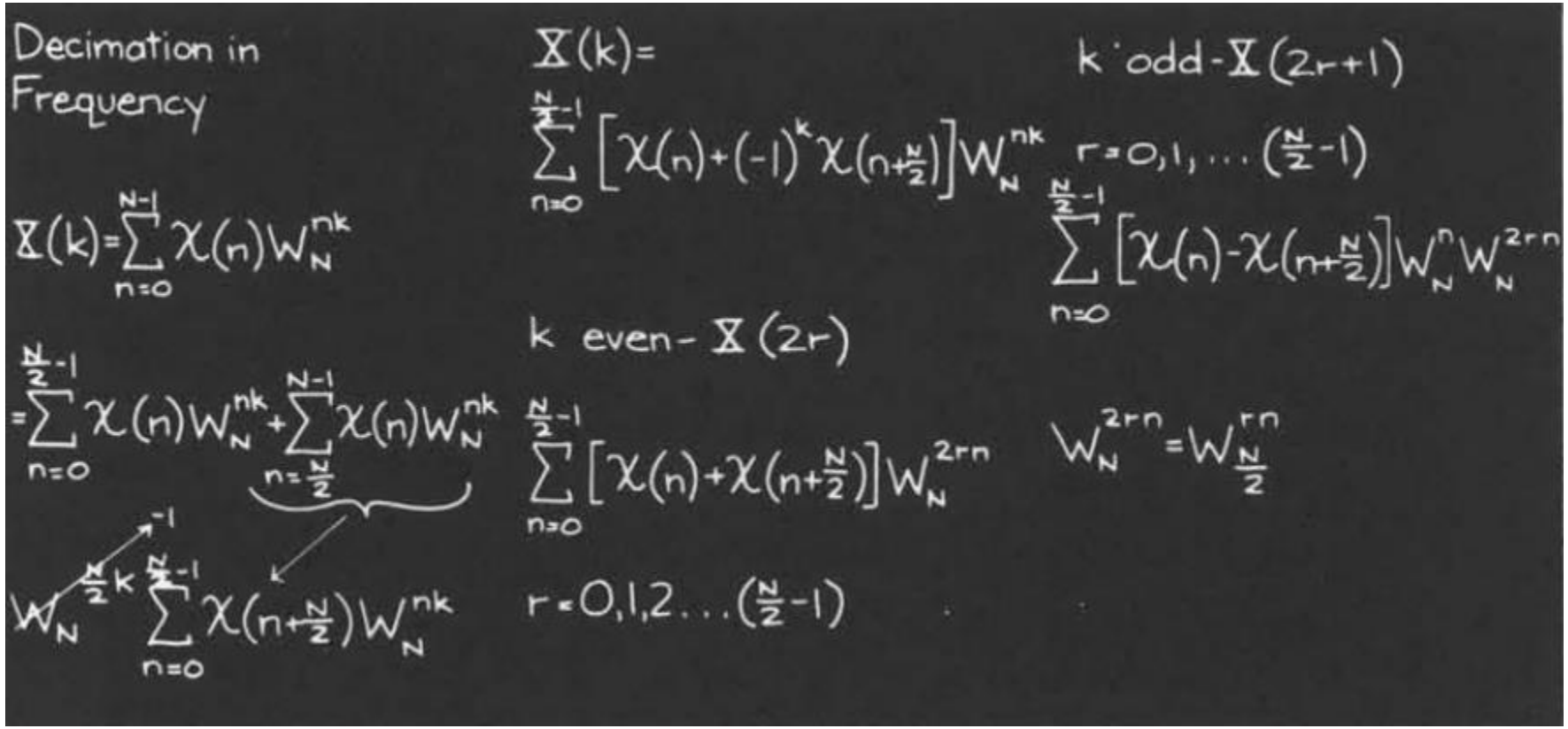
Rearrangement of the butterfly computation in 1.

Class Review

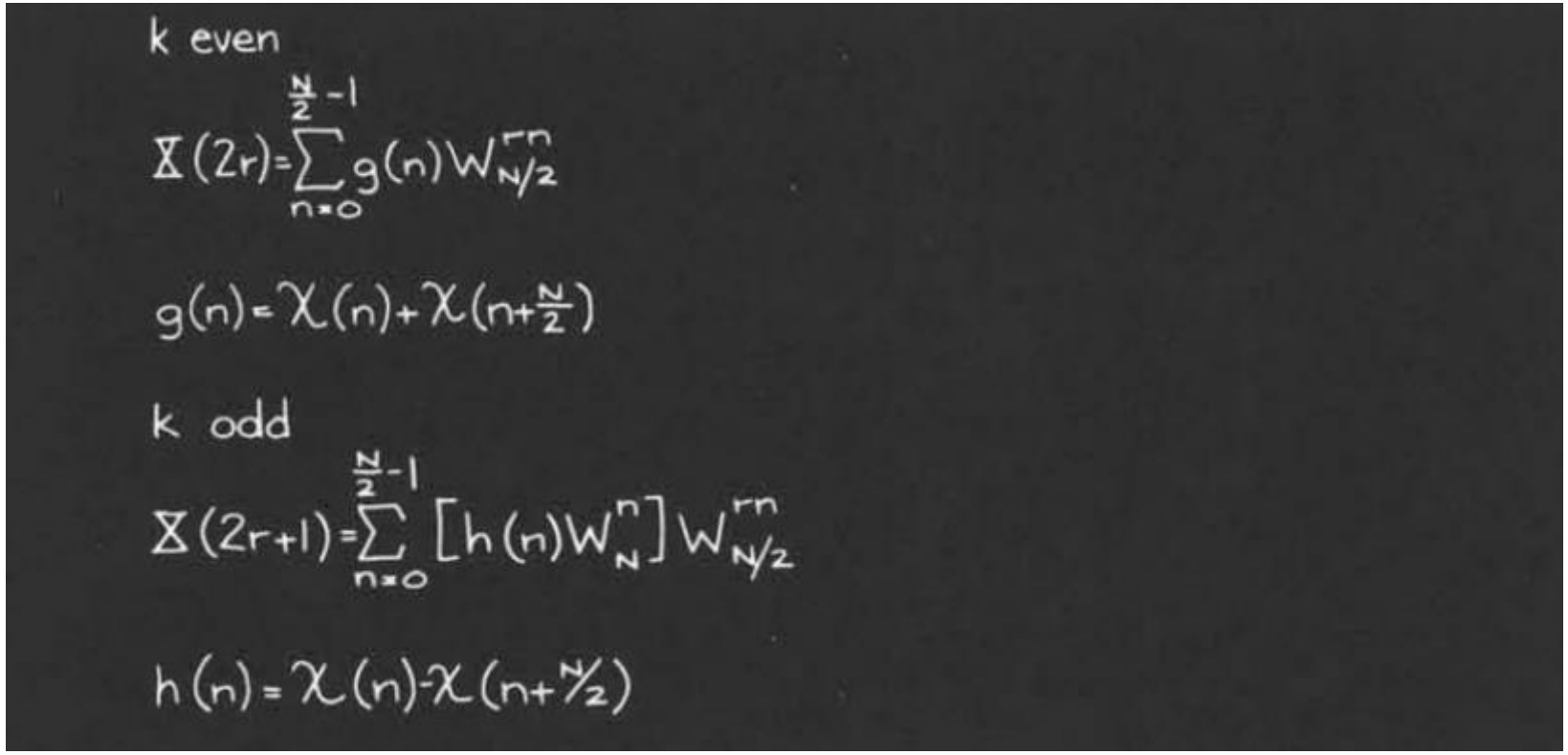


Decimation-in-time FFT algorithm utilizing the butterfly computation in m.

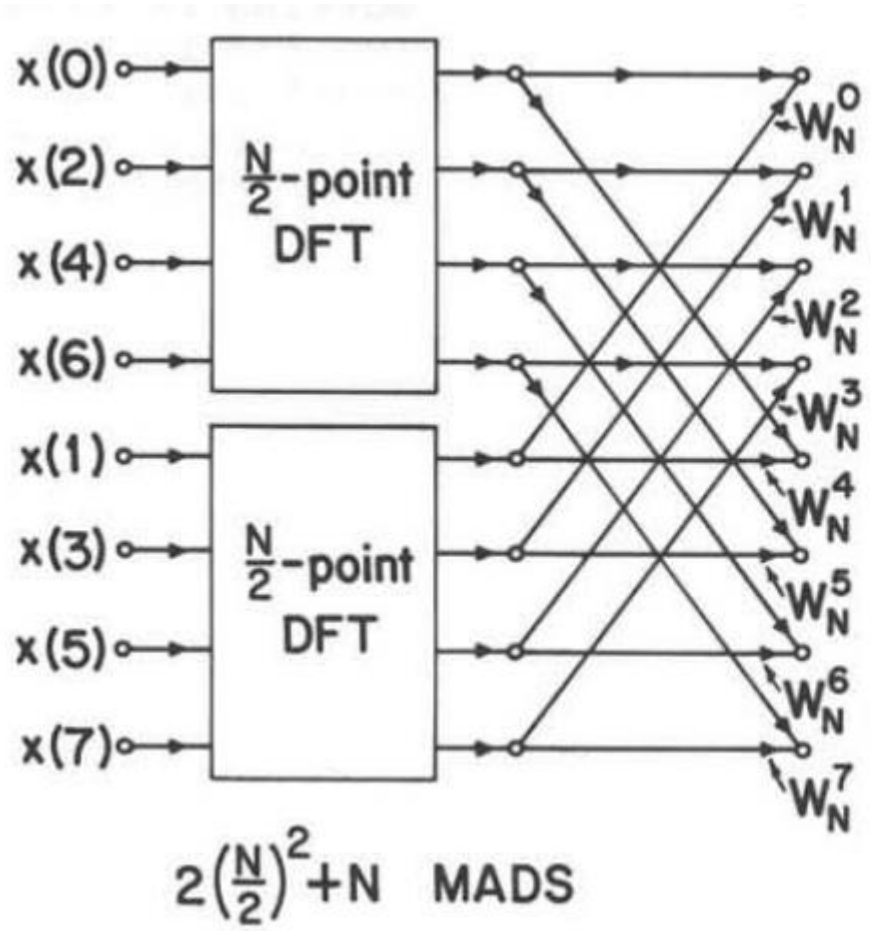
Class Review



Class Review

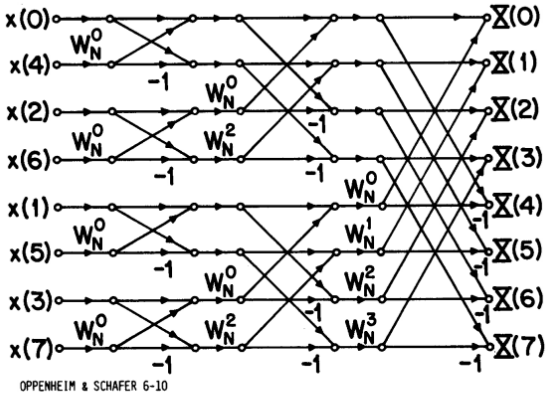


Class Review



Decomposition of an N-point DFT into 2 N/2-point DFT's.

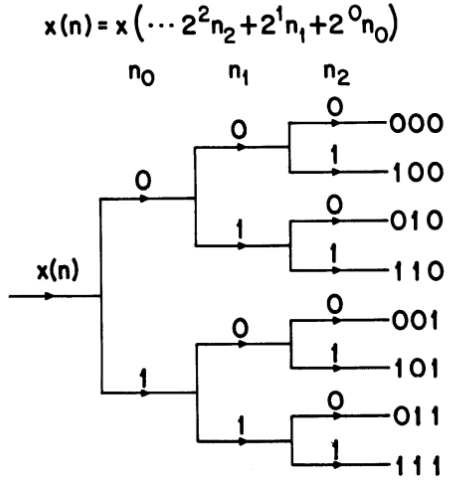
Class Review



FFT flow-graph for decimation-in-time algorithm.

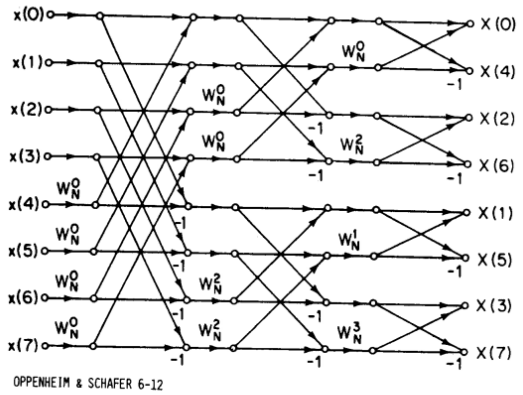
Storage Register		Data Index	
0	000	X(0)	000
1	001	X(4)	100
2	010	X(2)	010
3	011	X(6)	110
4	100	X(1)	001
5	101	X(5)	101
6	110	X(3)	011
7	111	X(7)	111

Relation between data index and data storage register.



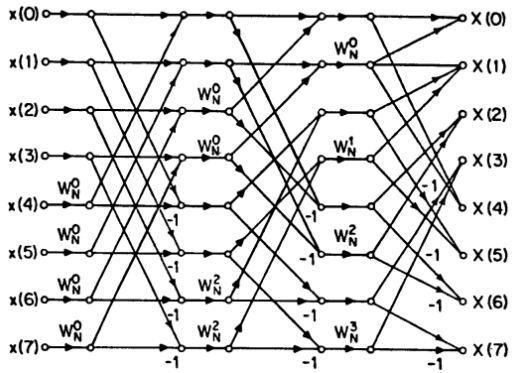
Sorting of data implied by the development of the decimation-in-time algorithm.

Class Review



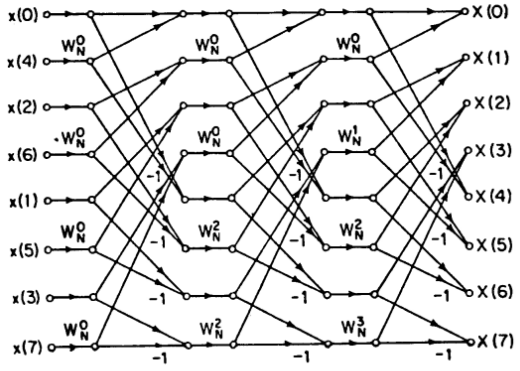
OPPENHEIM & SCHAFER 6-12

Rearrangement of flow-graph d. with data in normal order and output in bit-reversed order.



OPPENHEIM & SCHAFER 6-13

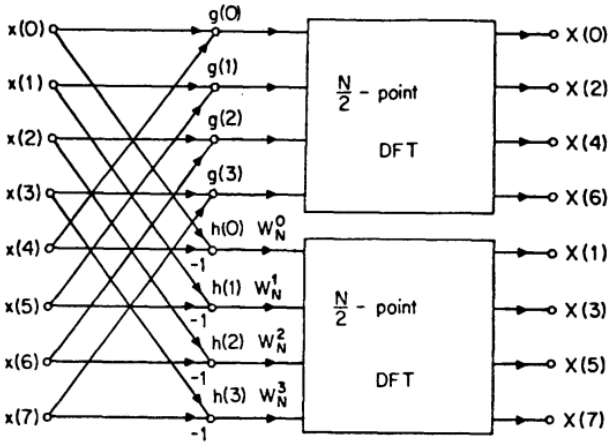
Rearrangement of flow-graph d. with both input and output in normal order.



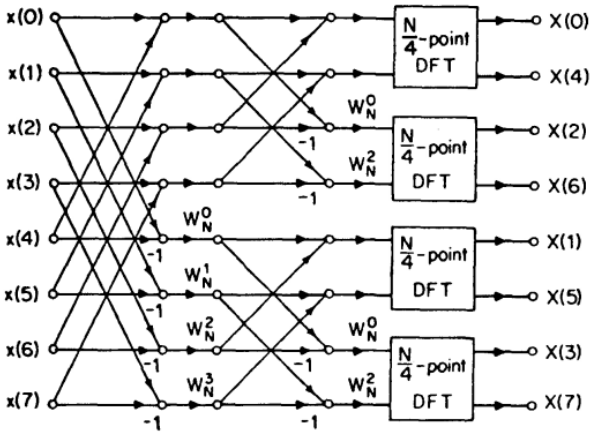
OPPENHEIM & SCHAFER 6-14

Rearrangement of flow-graph d having the same geometry for each stage.

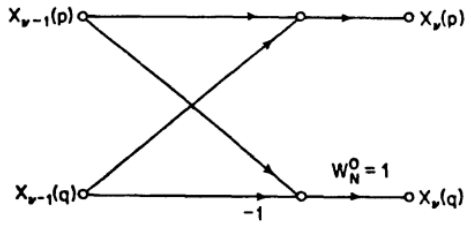
Class Review



Computation of even and odd-numbered DFT values.

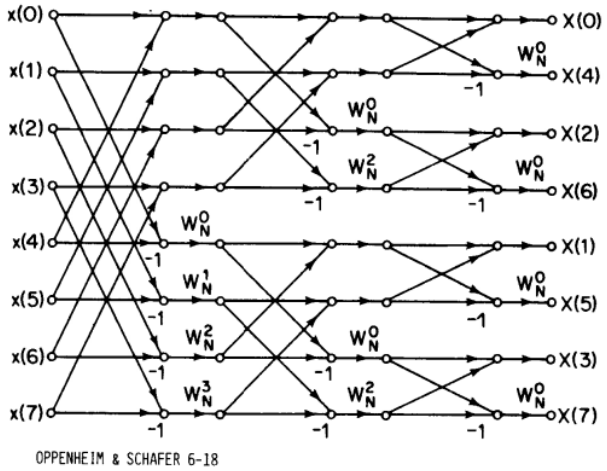


Decomposition of the N/2-point DFT's of flow-graph j.

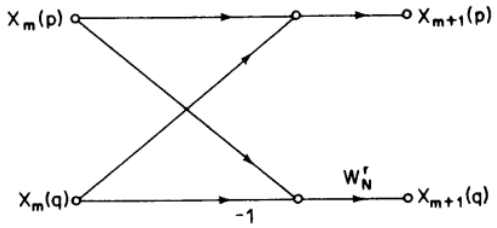


Flow-graph for two-point DFT.

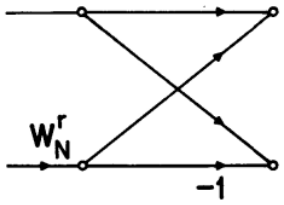
Class Review



Flow-graph of complete decimation-in-frequency decomposition of an eight-point DFT computation.

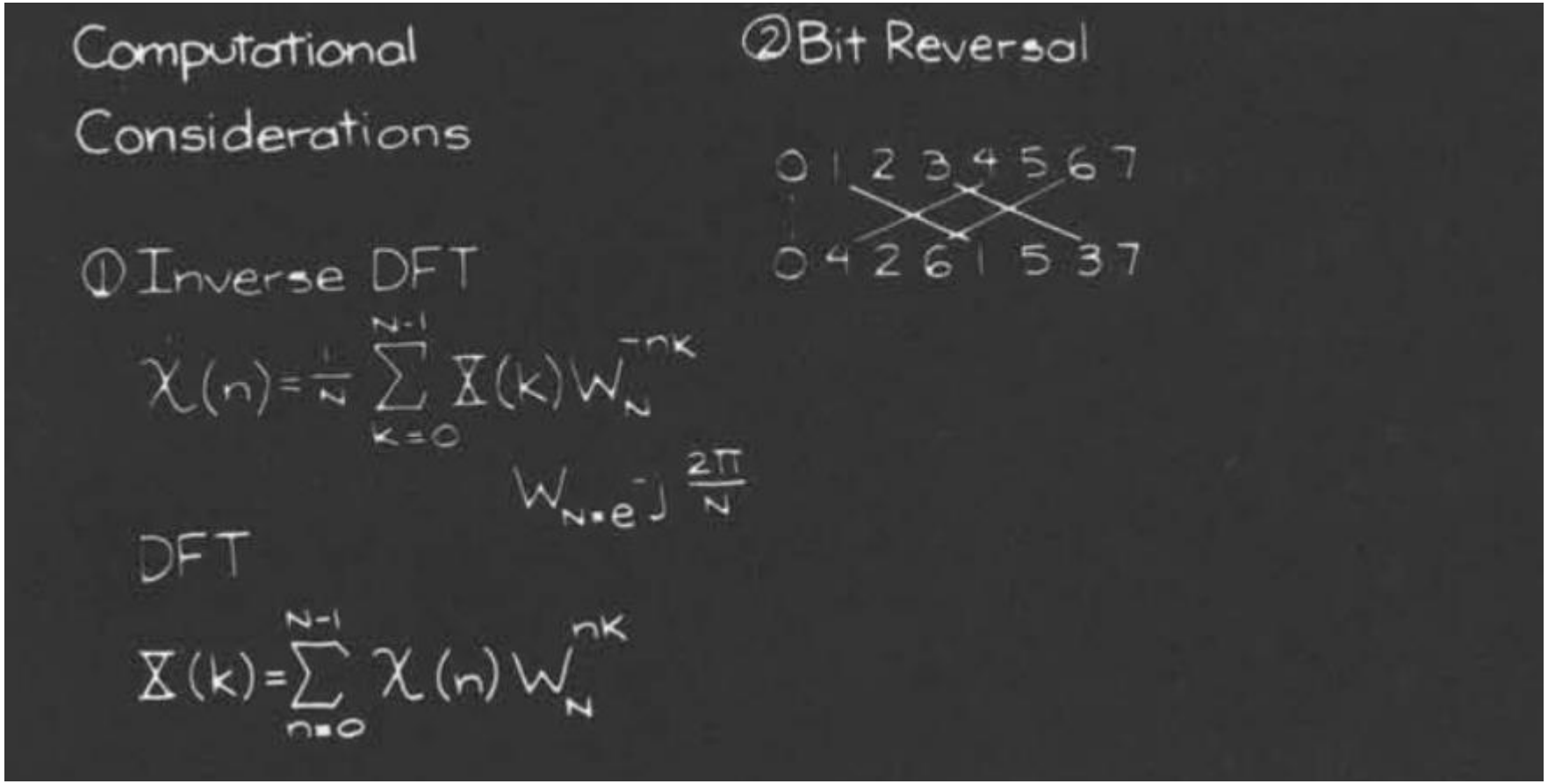


Flow-graph of a typical butterfly computation required in decimation-in-frequency FFT algorithm.



Flow-graph of a typical butterfly computation required in decimation-in-time FFT algorithm.

Class Review



Class Review

$N = P_1 \cdot \underbrace{P_2 \cdots P_v}_{q_1}$

$N = P_1 \cdot q_1$

Subsequence: every P_1^{th} point

P_1 sequences of length q_1

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}$$

$$= \sum_{r=0}^{q_1-1} x(P_1 r) W_N^{P_1 rK}$$

$$+ \sum_{r=0}^{q_1-1} x(P_1 r+1) W_N^{(P_1 r+1)k}$$

$$+ \cdots$$

$$X(k) =$$

$$\sum_{l=0}^{P_1-1} W_N^{lk} \sum_{r=0}^{q_1-1} x(P_1 r+l) W_N^{P_1 rk}$$

$$W_N^{P_1 rk} = e^{-j\frac{2\pi}{P_1 \cdot q_1} P_1 rk}$$

$$= e^{-j\frac{2\pi}{q_1} rk} = W_{q_1}^{rk}$$

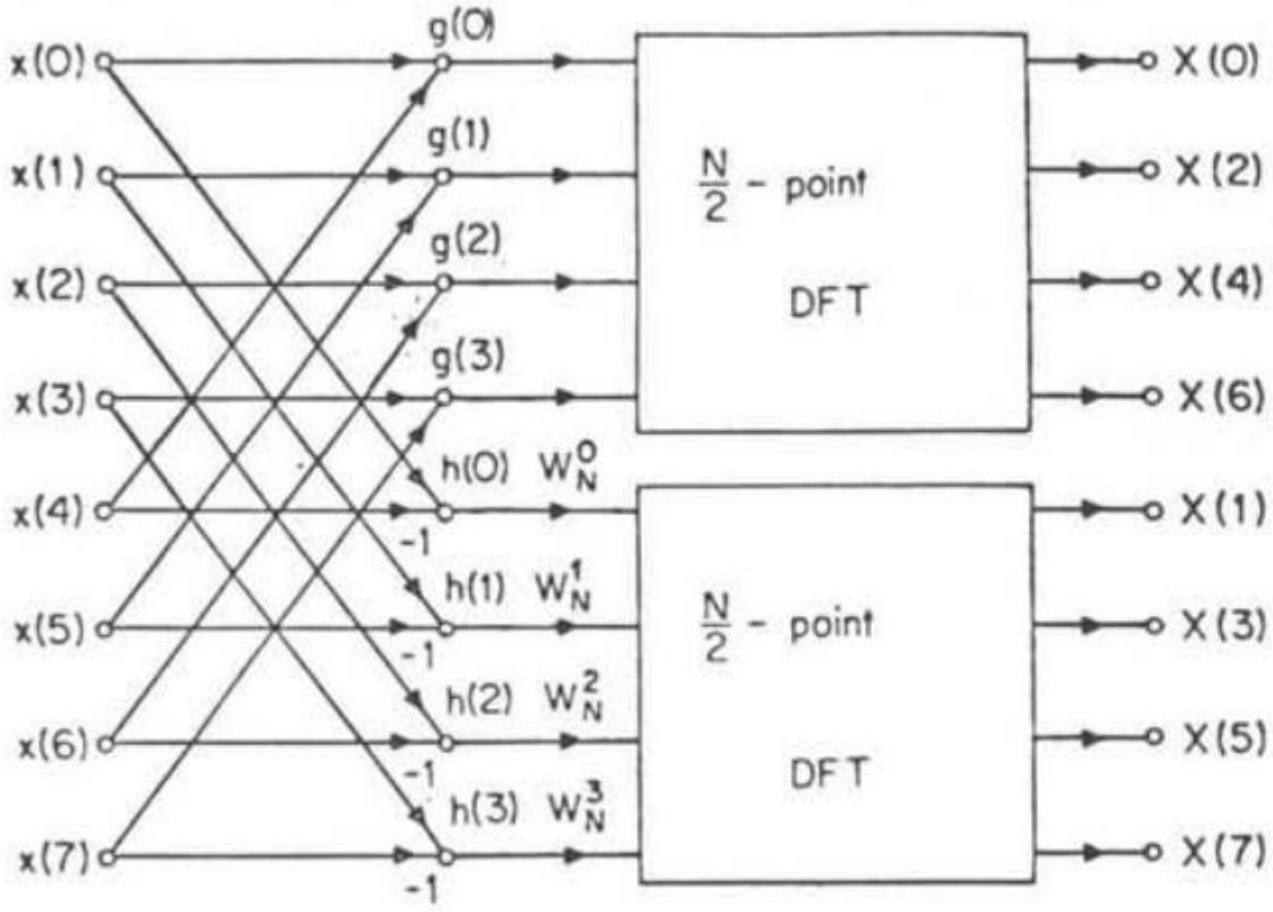
$$X(k) = \sum_{l=0}^{P_1-1} W_N^{lk} \underbrace{\sum_{r=0}^{q_1-1} x(P_1 r+l) W_{q_1}^{rk}}_{q_1\text{-point DFT}}$$

$q_1 = P_2 \cdot \underbrace{P_3 \cdots P_v}_{q_2}$

MADS $\Rightarrow$

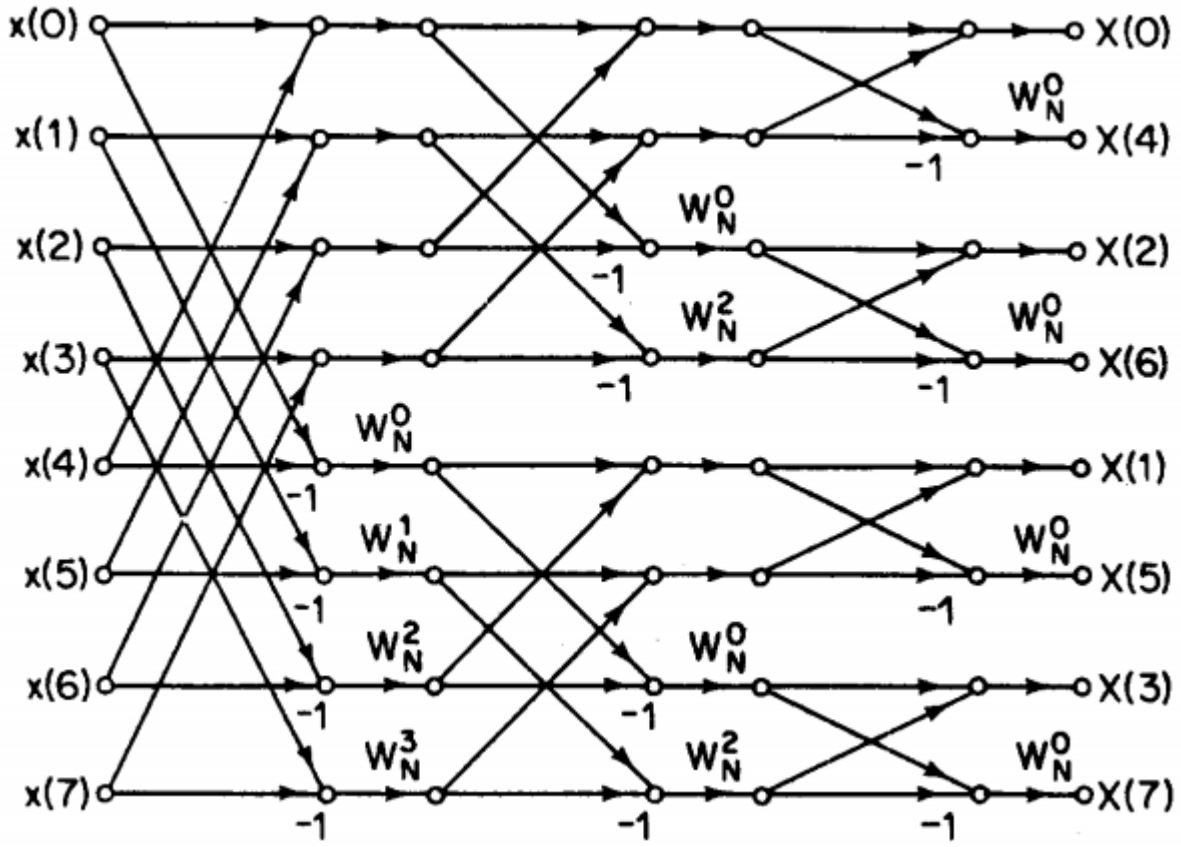
$N[P_1 + P_2 + \cdots + P_v - v]$

Class Review



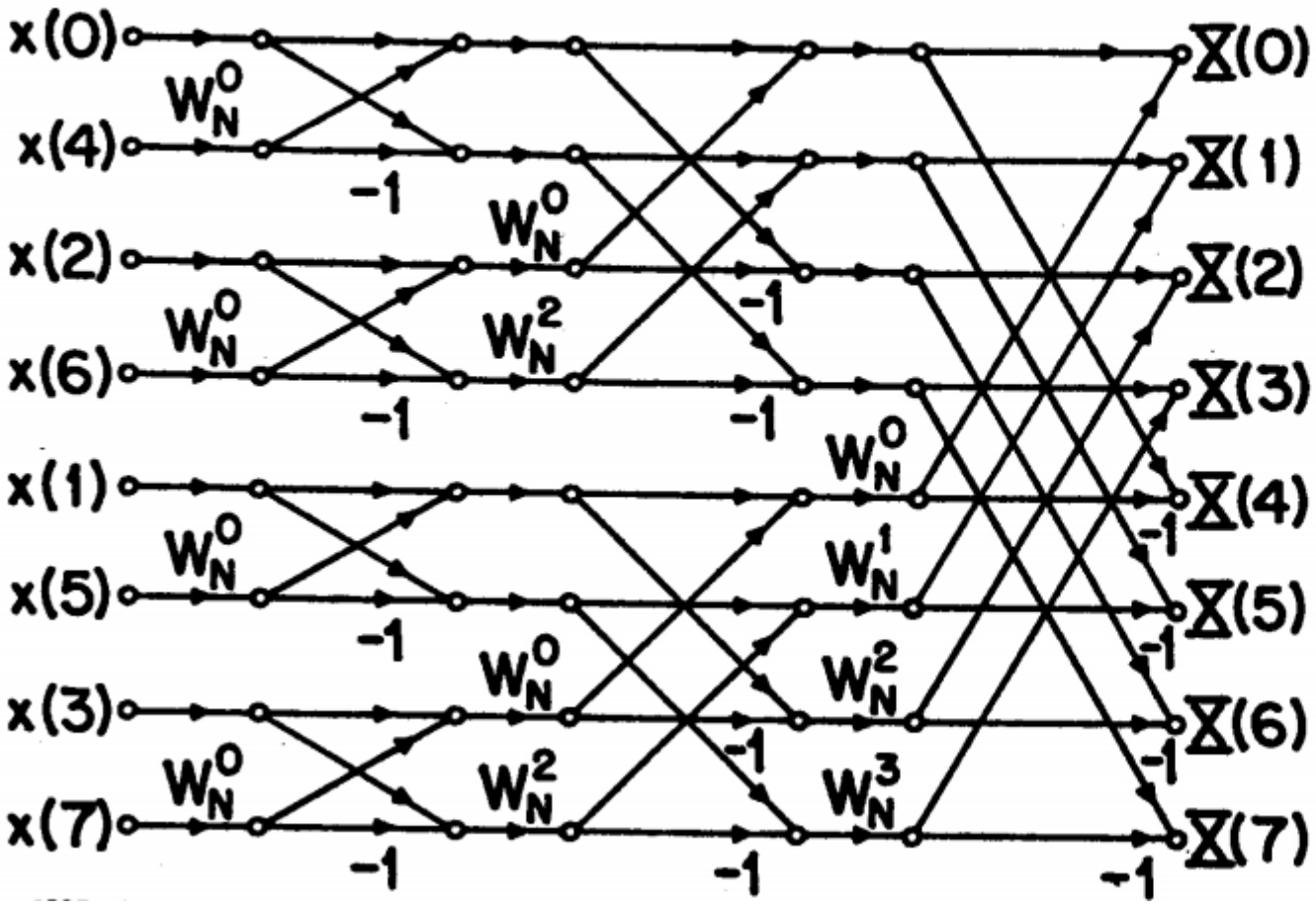
Computation of even and odd numbered DFT values.

Class Review



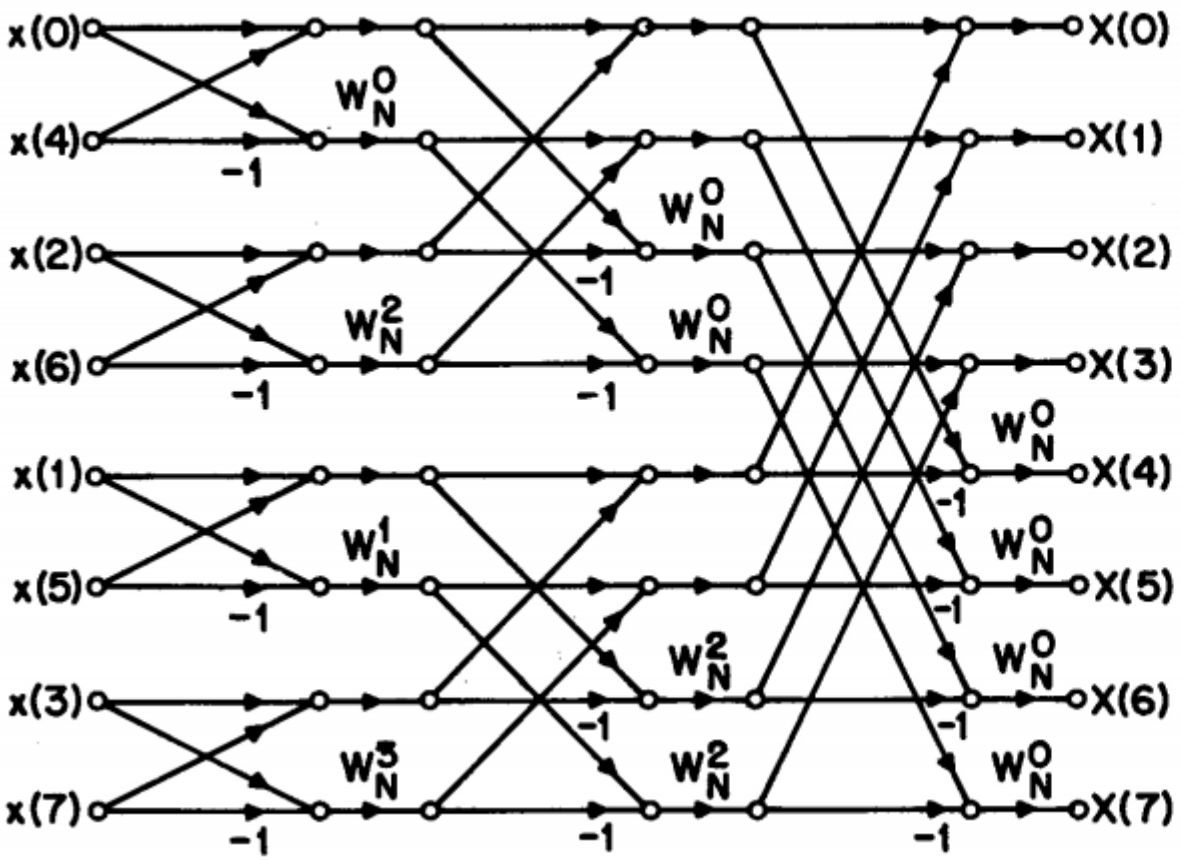
Flow-graph of complete decimation-in-frequency decomposition of an eight point DFT computation.

Class Review



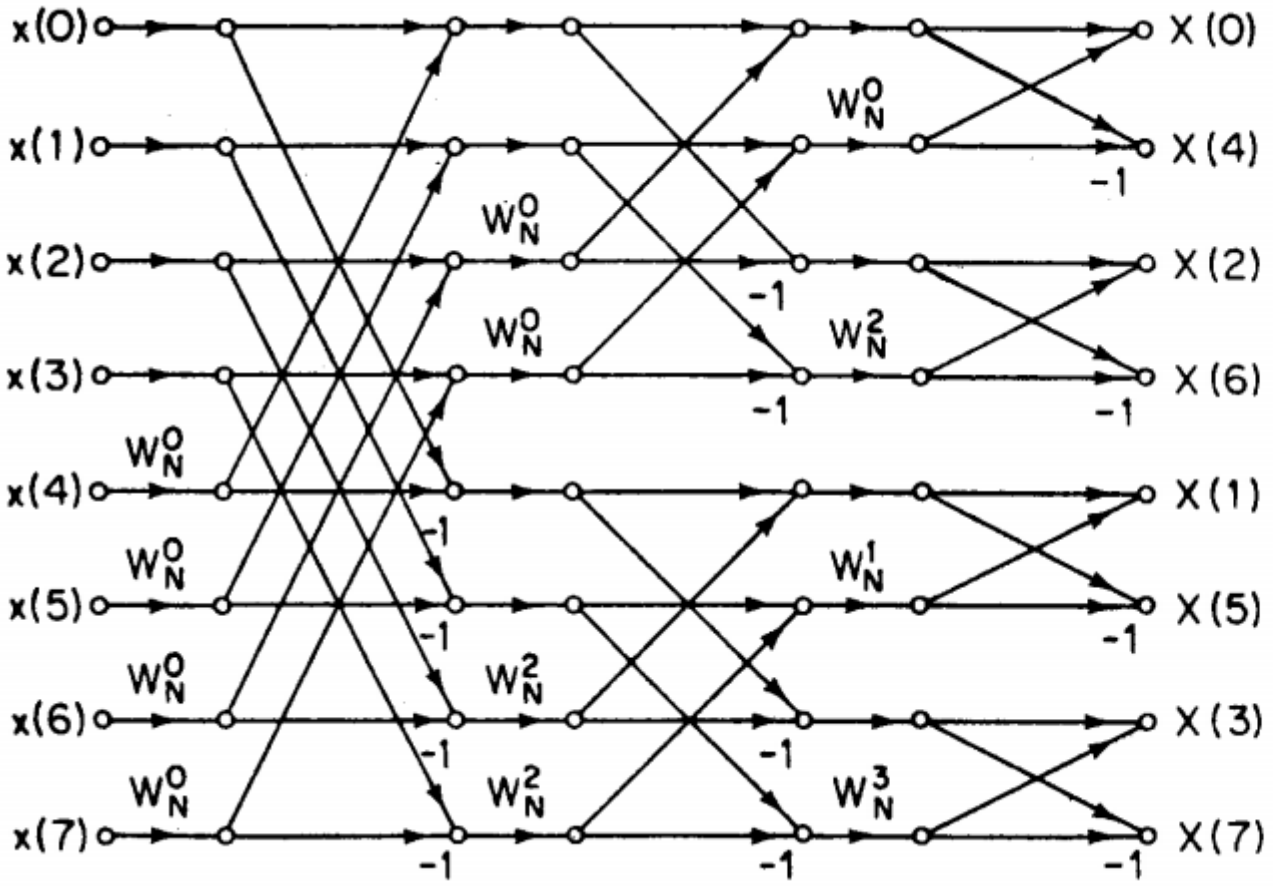
FFT flow-graph for decimation-in-time algorithm.

Class Review



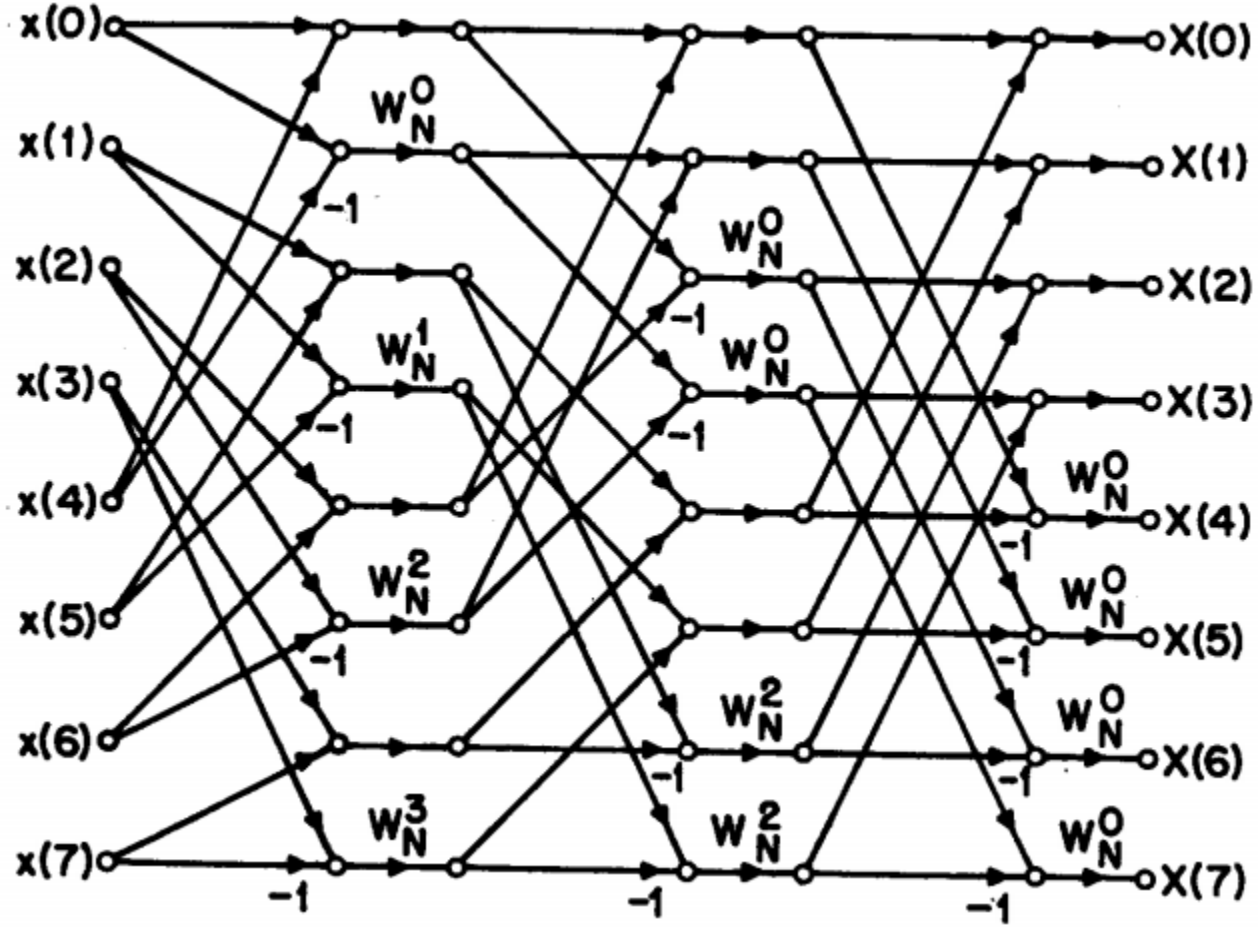
Rearrangement of the decimation-in-frequency flow-graph d. The input is now in bit-reversed order and the output is in normal order.

Class Review



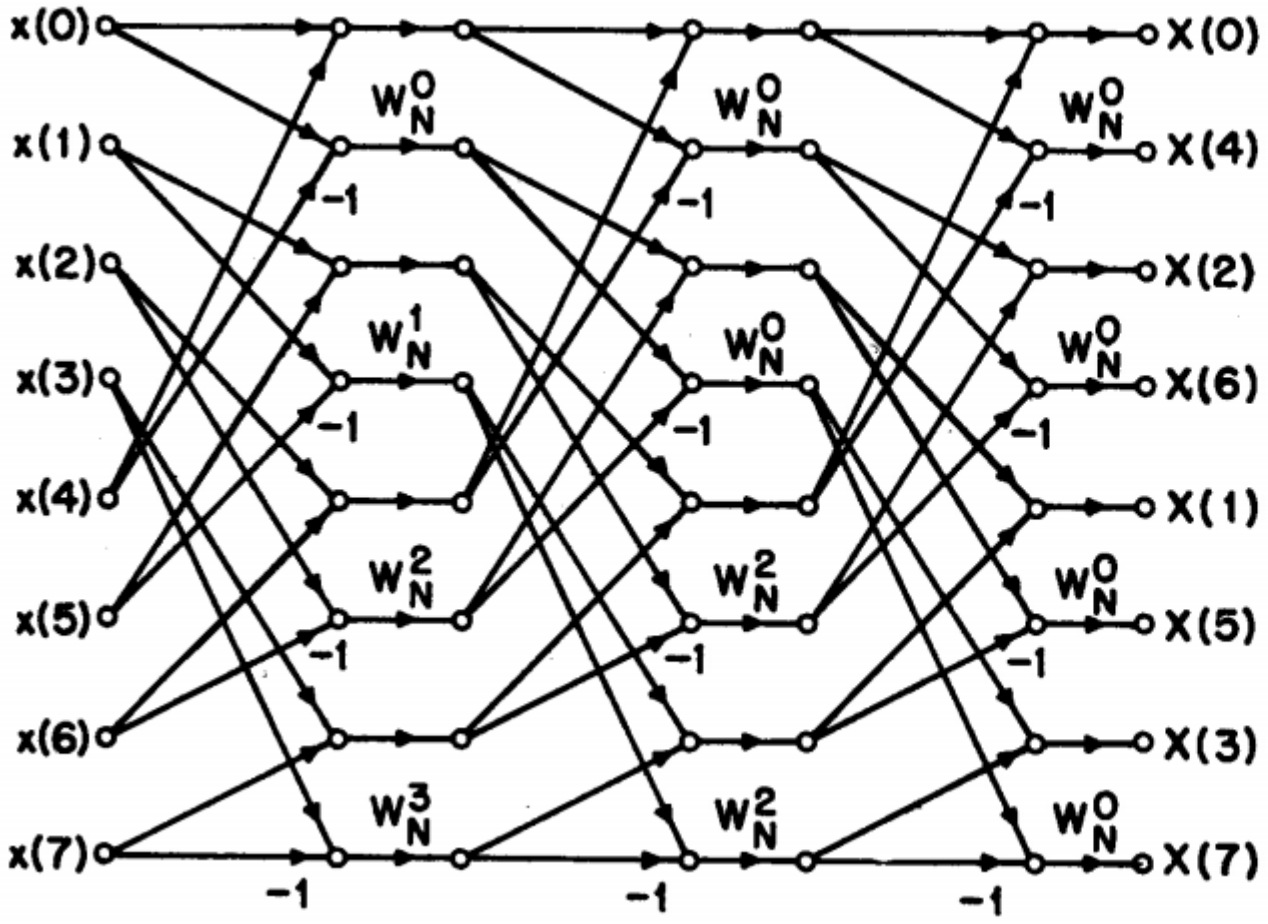
Rearrangement of e so that the input is in normal order and output in bit-reversed order.

Class Review



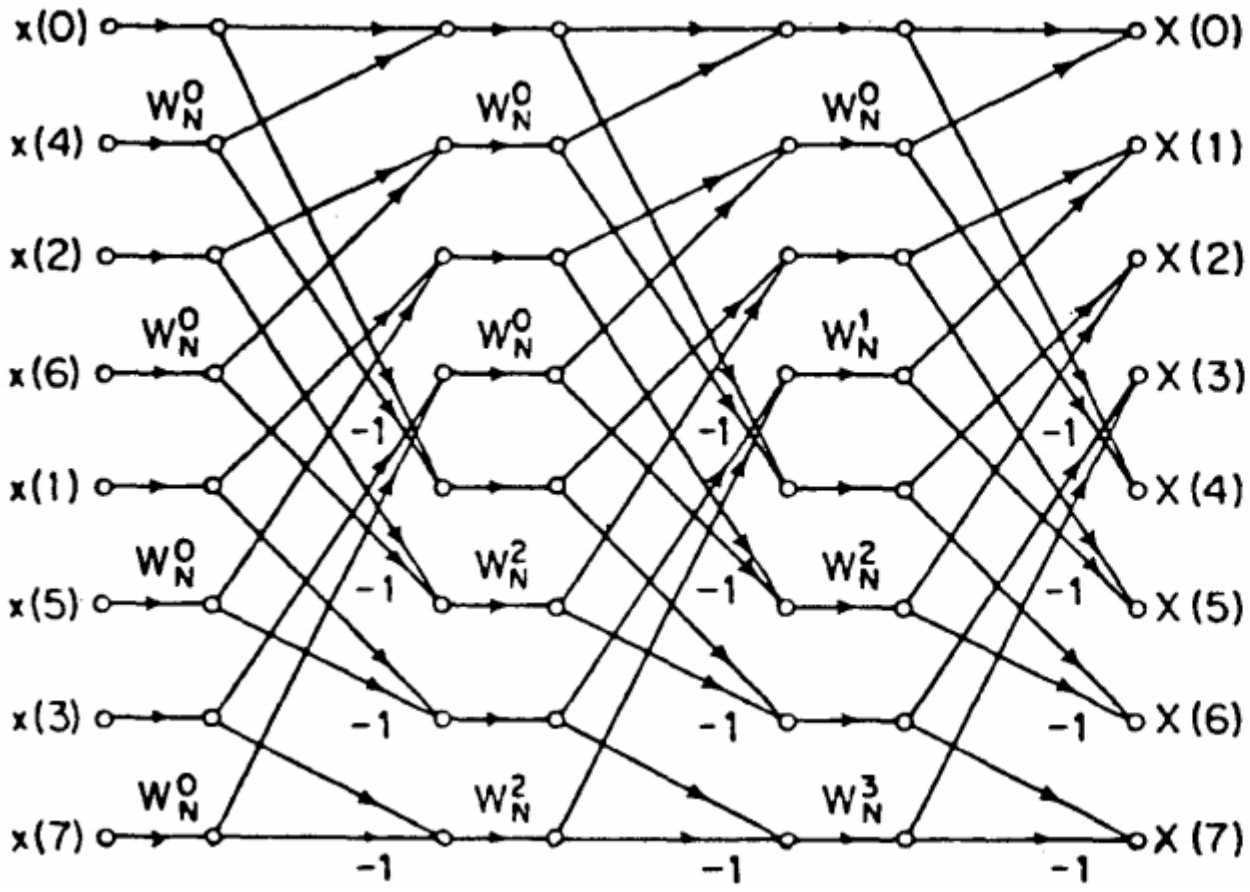
Rearrangement of d so that both input and output are in normal order.

Class Review



Rearrangement of d so that geometry is identical in each stage.

Class Review



Decimation-in-time flow-graph for which the geometry is identical for each stage.