



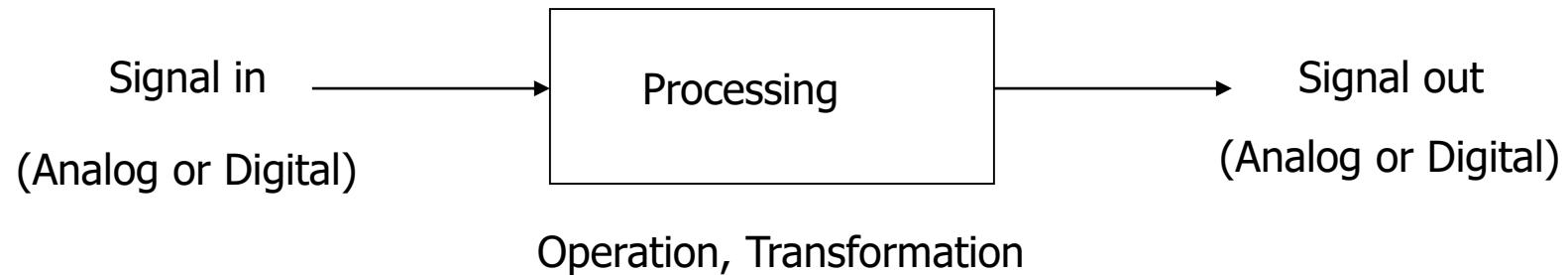
# Digital Signal Processing

Arak University of Technology

By: Dr. Moein Ahmadi



# What is Signal Processing?



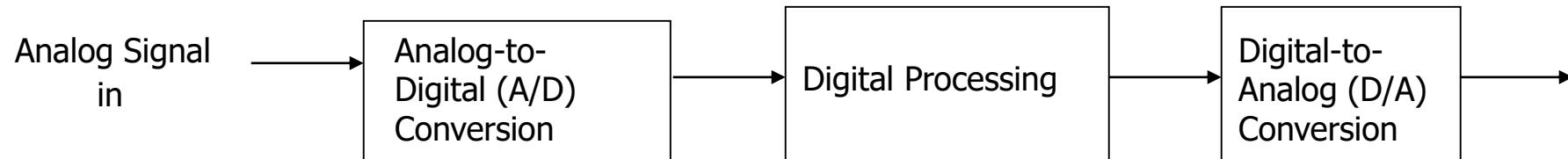
- Example of Signals:
  - Analog: Speech, Music, Photos, Video, Radar, Sonar, ...
  - Discrete-domain/Digital:
    - digitized speech, digitized music, digitized images, digitized video, digitized radar and sonar signals,...
    - stock market data, daily max temperature data, ...

# Digital Signal Processing and ADC

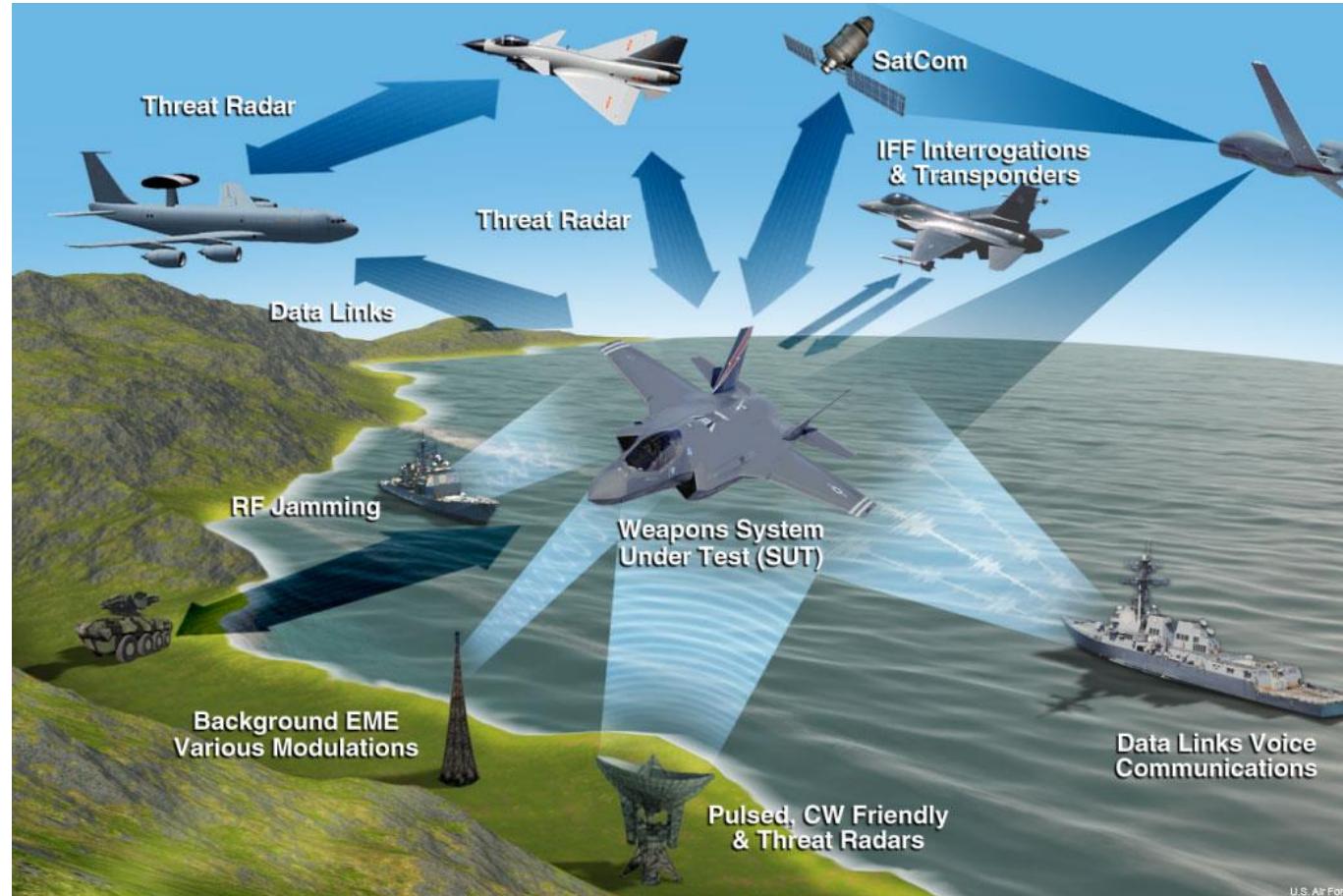


Operation, Transformation performed  
on digital signals (using a computer or  
other special-purpose digital hardware)

- But what about analog signals?



# General System- EW Environment



رادار  
شنود  
جمینگ  
مخابرات  
سنسور های مختلف

رادار جستجو - رهگیر - تصویر برداری

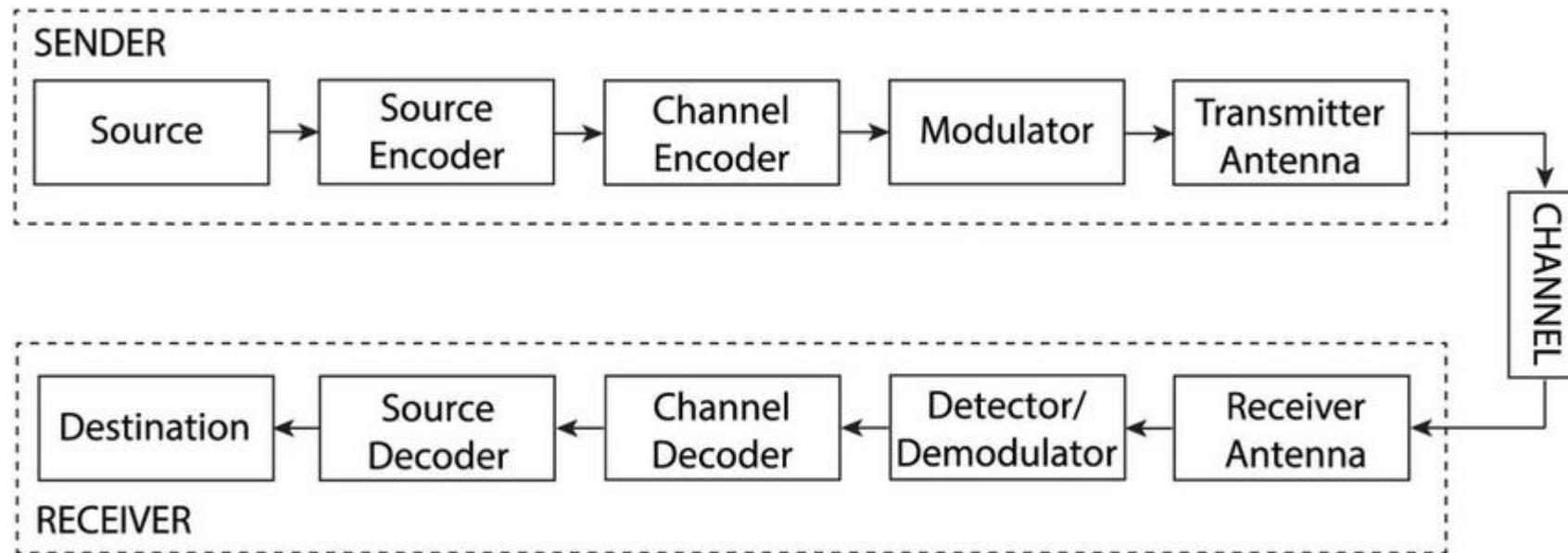
رادار جستجو:

آشکارسازی و تعیین فاصله و سرعت اهداف

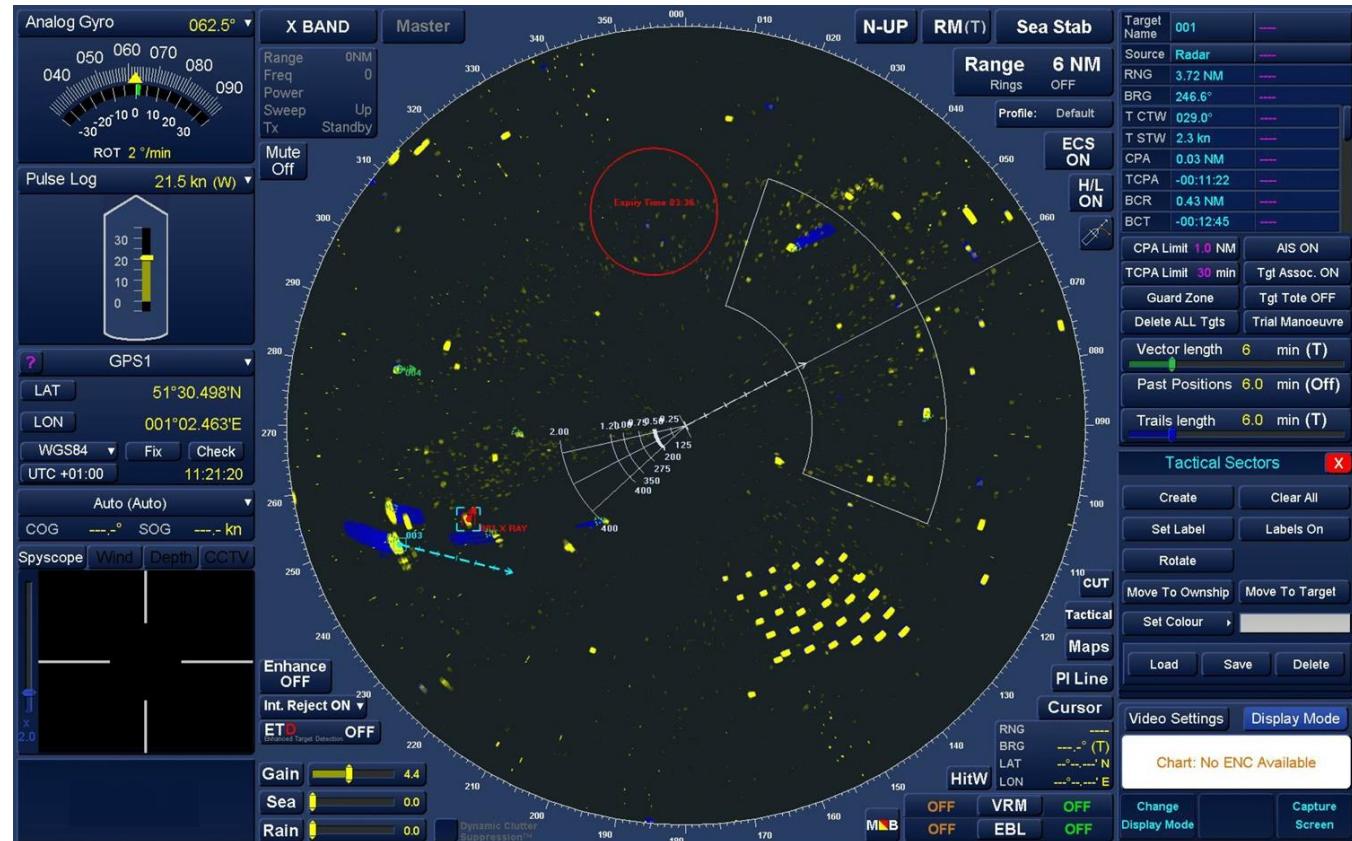
فیلتر فشرده سازی  
پردازش داپلر  
آشکارسازی  
رهگیری حین اسکن

نمونه هایی از سیستم های مورد بحث  
در پردازش سیگنال

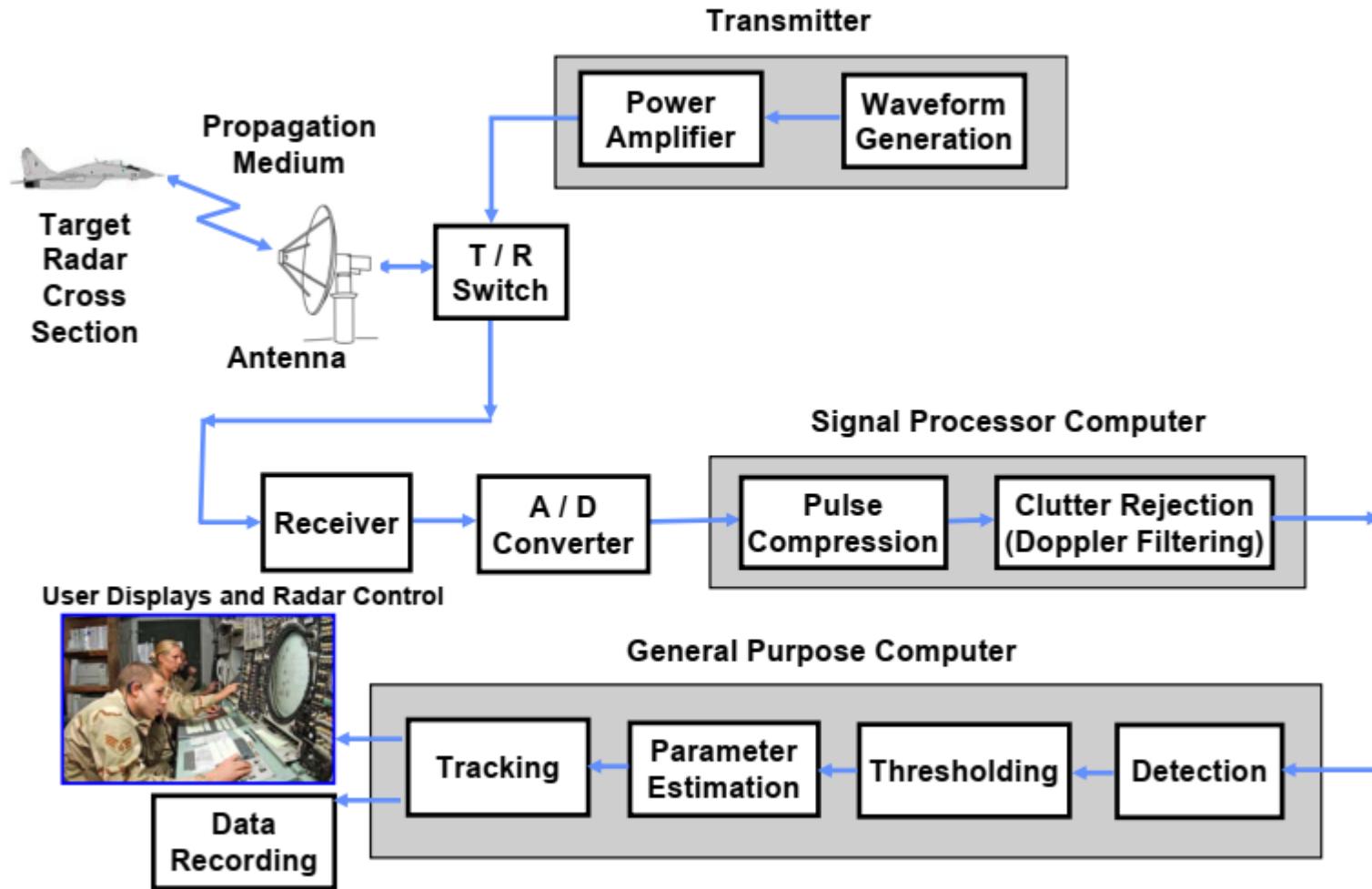
# Communications System Block Diagram



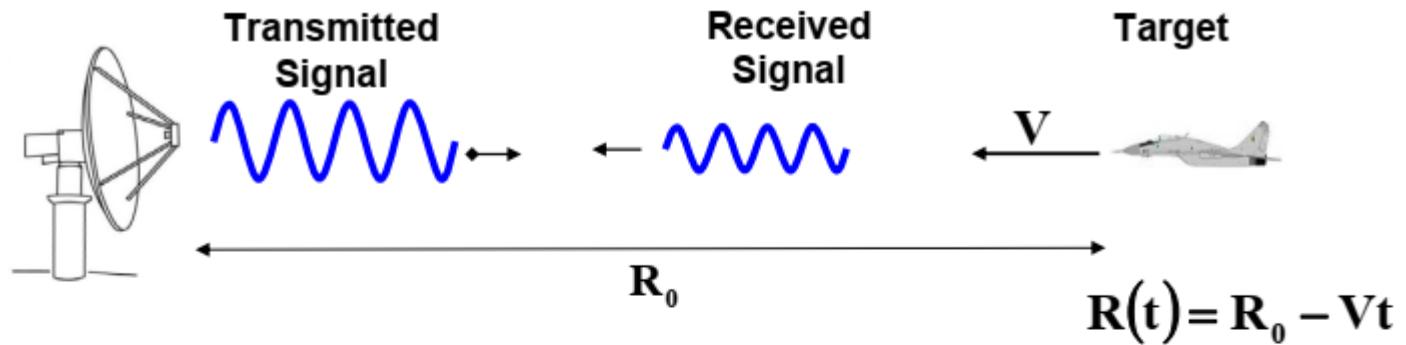
# General System- Phased Array Radar



# Example- Radar System



# Example- Radar Basics



**Transmitted Signal:**

$$s_T(t) = A(t) \exp(j 2\pi f_0 t)$$

**Received Signal:**

$$s_R(t) = \alpha A(t - \tau) \exp[j 2\pi (f_0 + f_D)t]$$

### Amplitude

Depends on RCS, radar parameters, range, etc.

### Angle

Azimuth and Elevation

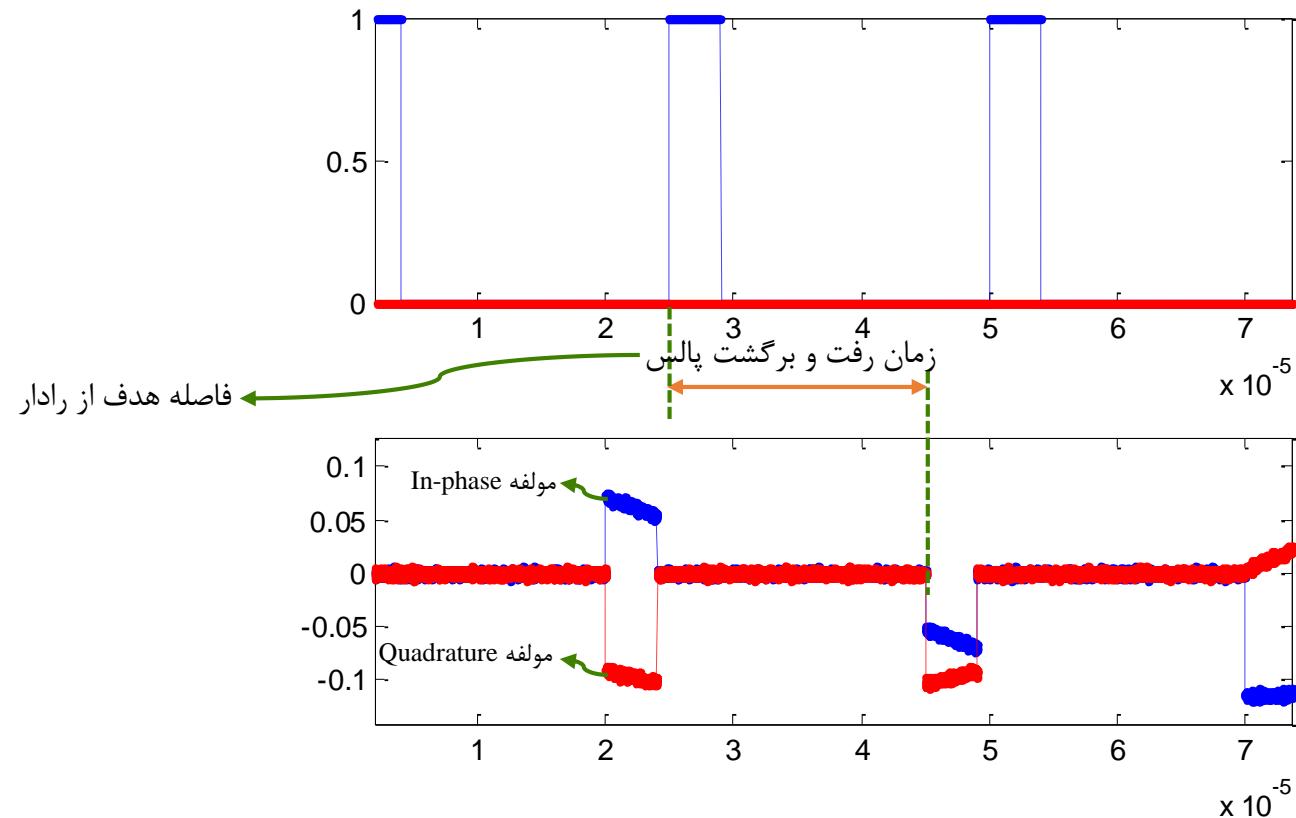
### Time Delay

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

### Doppler Frequency

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

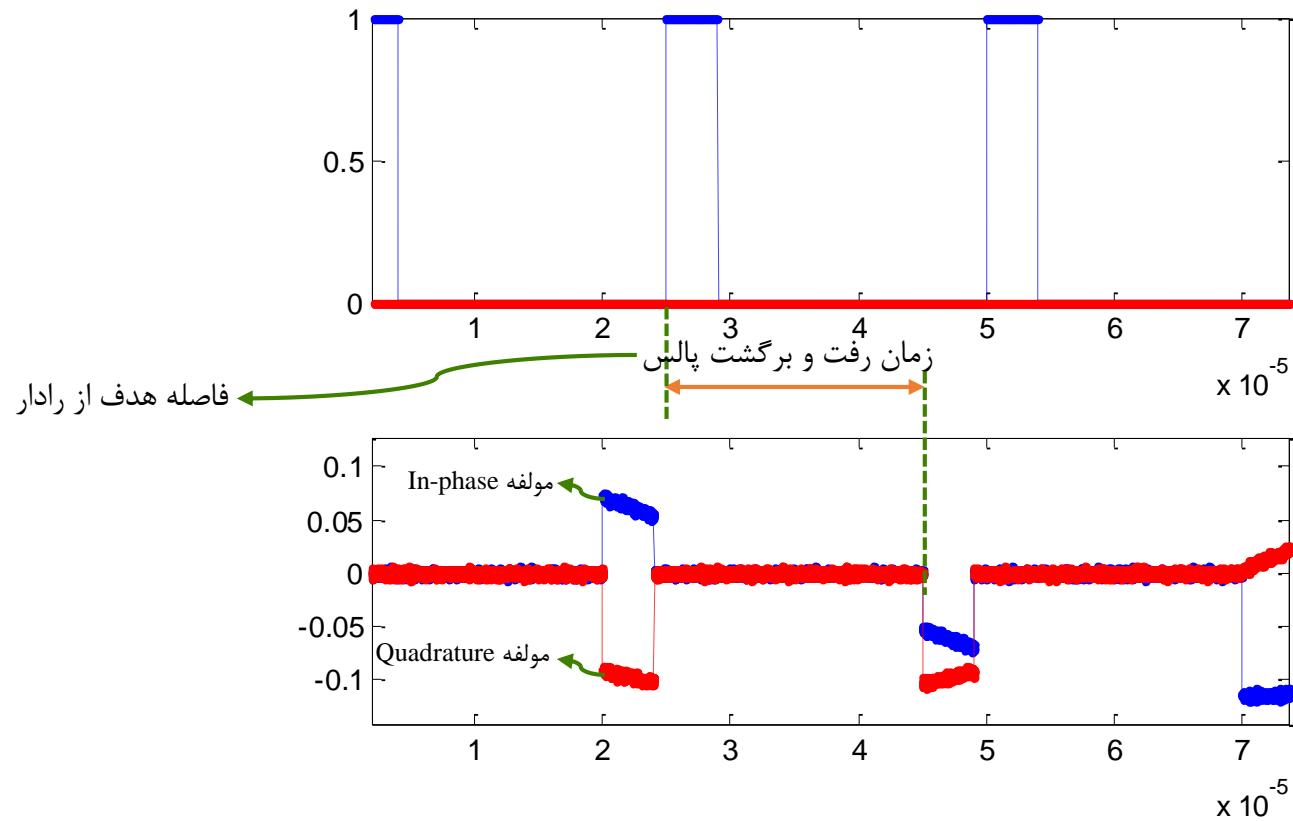
# Radar Basics



سیگنال ارسالی  
از فرستنده رادار

سیگنال دریافتی از  
هدف در گیرنده رادار

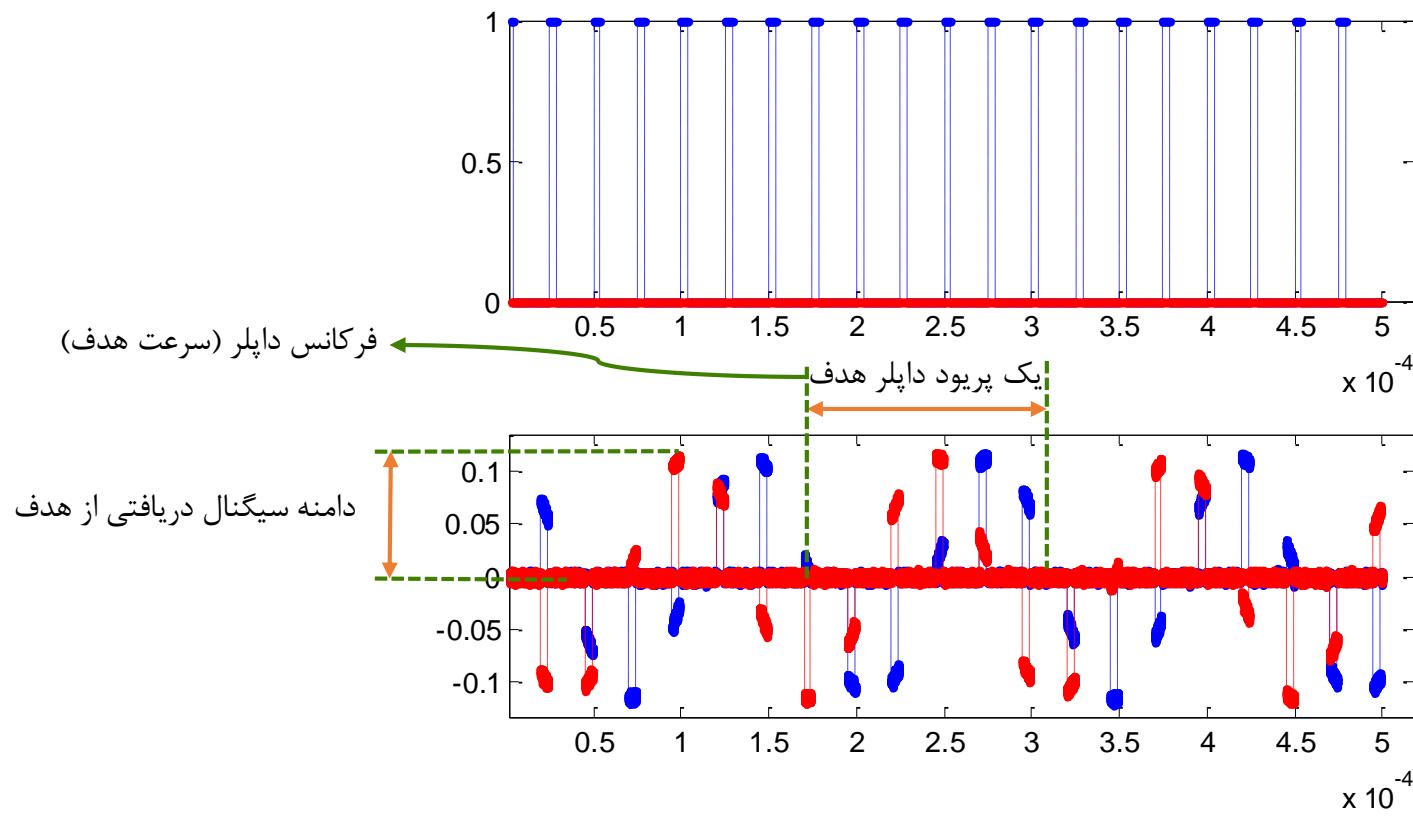
# Radar Basics



سیگنال ارسالی  
از فرستنده رادار

سیگنال دریافتی از  
هدف در گیرنده رادار

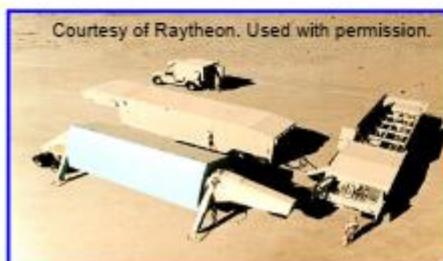
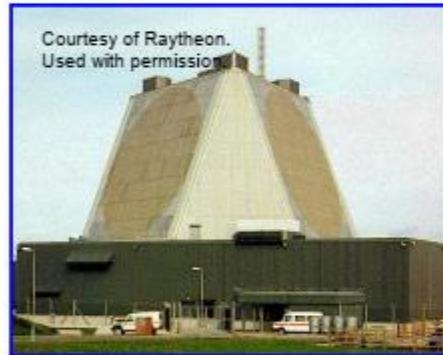
# Radar Basics



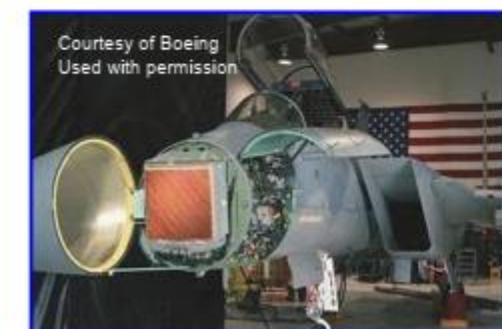
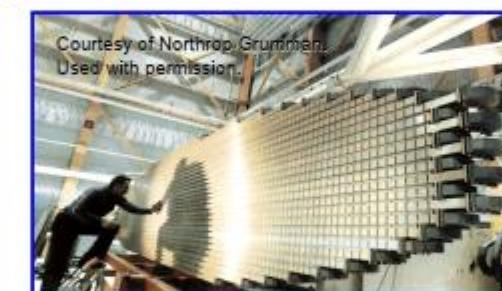
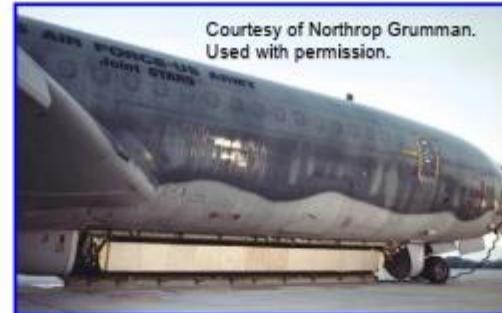
سیگنال ارسالی  
از فرستنده رادار

سیگنال دریافتی از  
هدف در گیرنده رادار

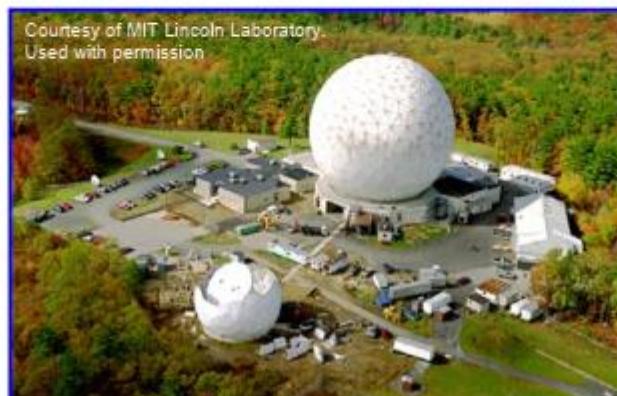
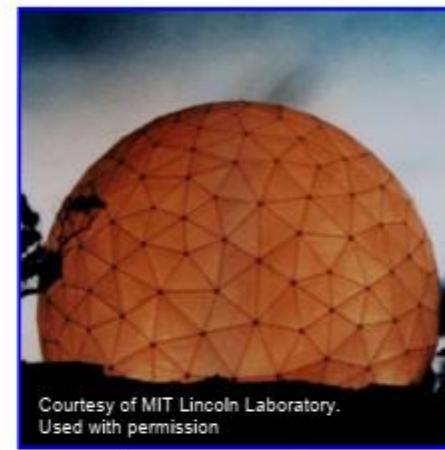
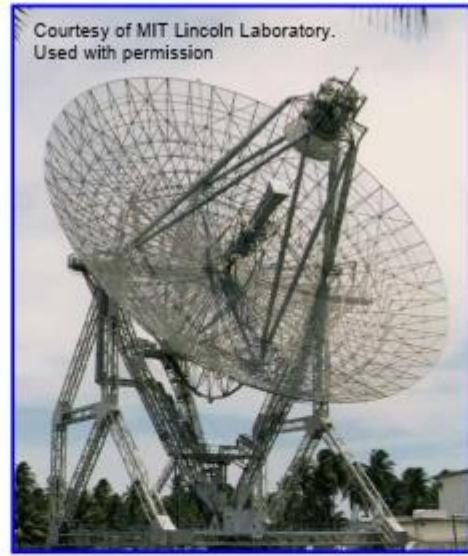
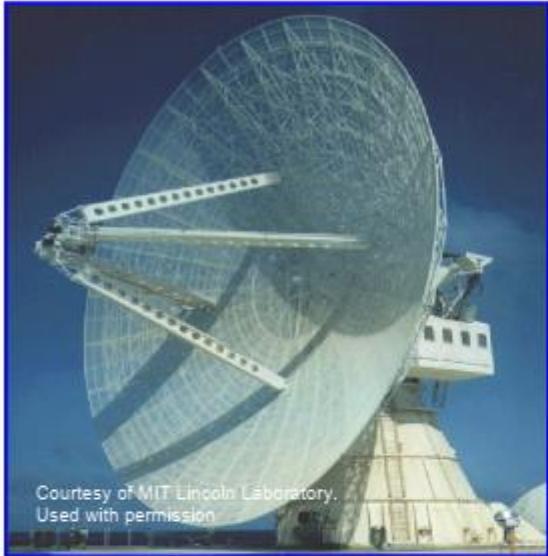
# Surveillance and Fire Control Radars



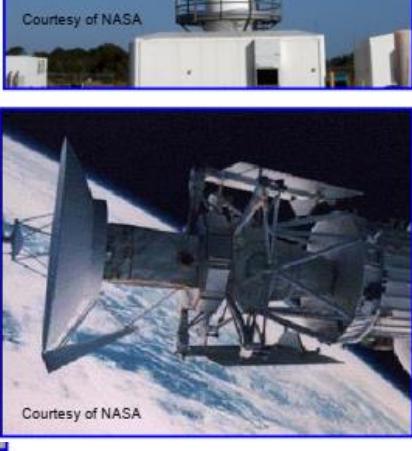
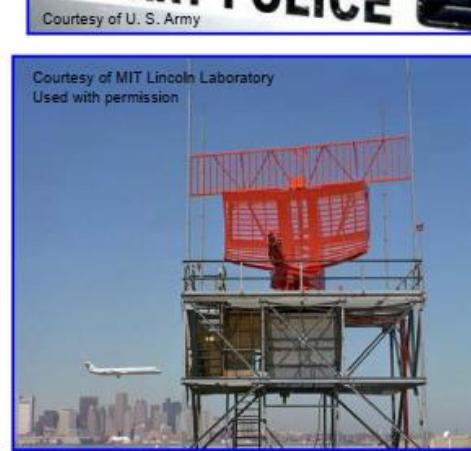
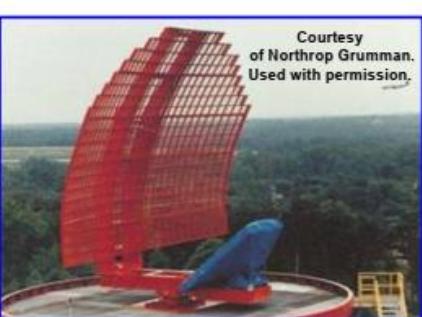
# Airborne Radars



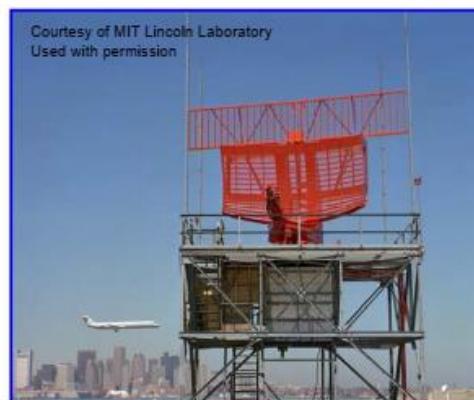
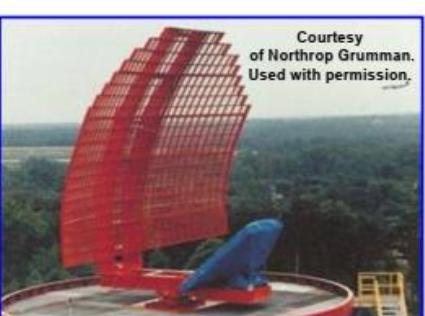
# Instrumentation Radars



# 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



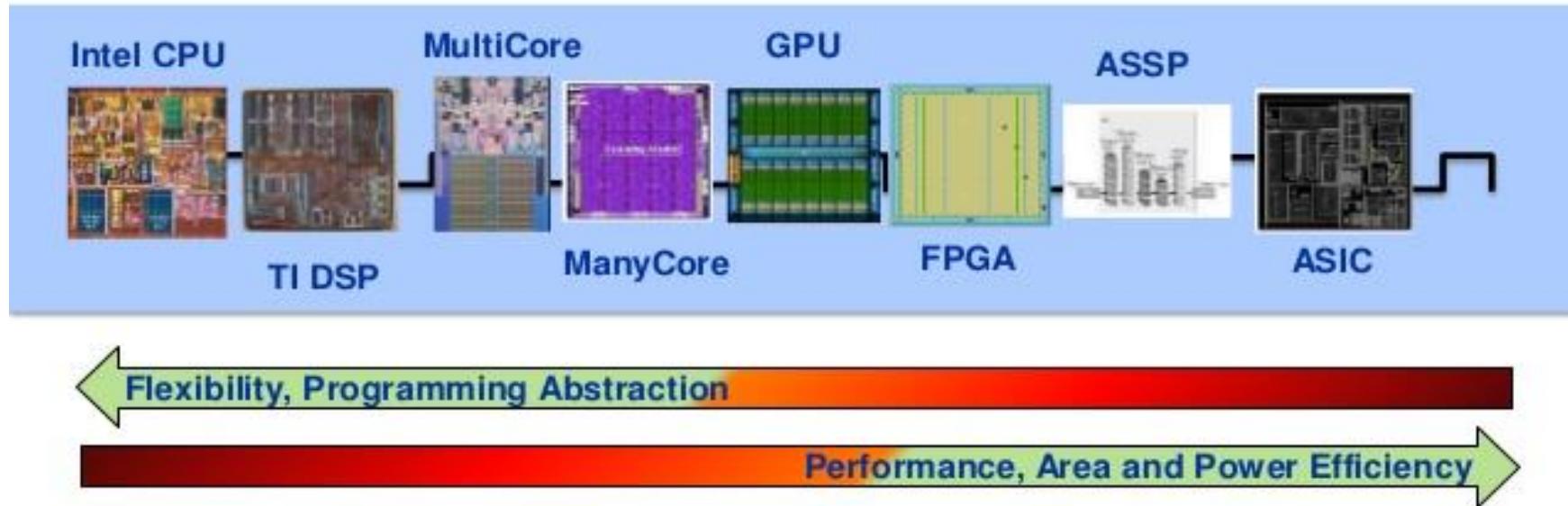
# Signal Processing Topics

- 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



## 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<sup>TM</sup>

MATLAB



Qt



jupyter    anaconda<sup>®</sup>    spyder



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**

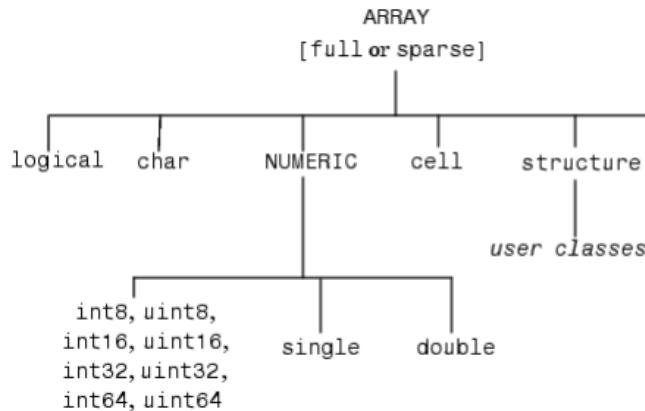
**Classes**

Libraries      Numpy      SciPy  
Matplotlib

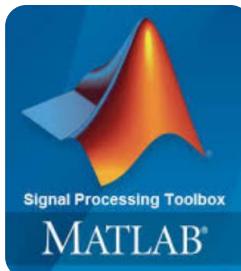


<https://www.mathworks.com>

**Basic data types:**



**Toolbox**



C++

<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**





## Getting Started With Python Using Anaconda



Windows



macOS



Linux

### Anaconda 2018.12 for Windows Installer

#### Python 3.7 version

[Download](#)

[64-Bit Graphical Installer \(614.3 MB\)](#)

[32-Bit Graphical Installer \(509.7 MB \)](#)

#### Python 2.7 version

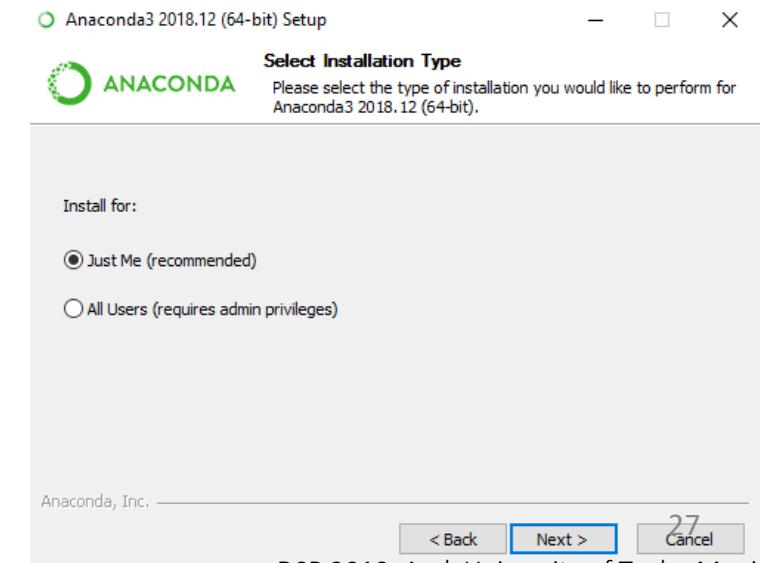
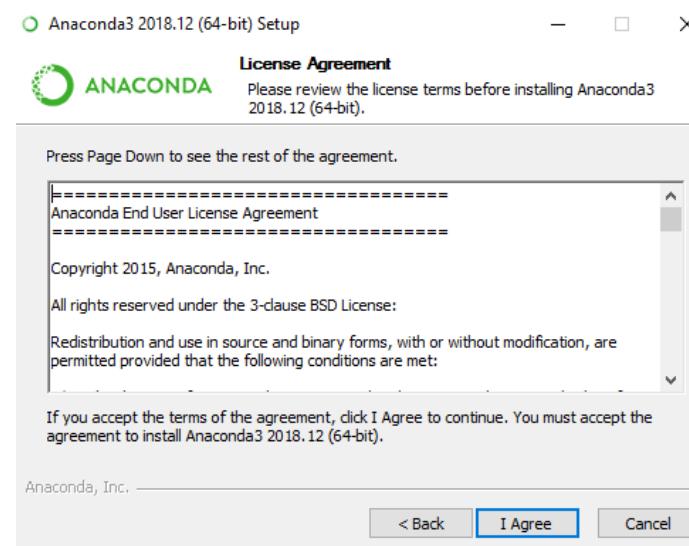
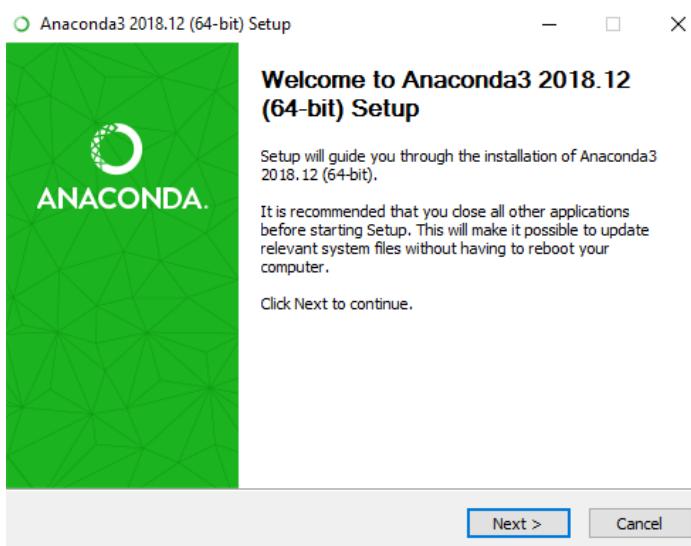
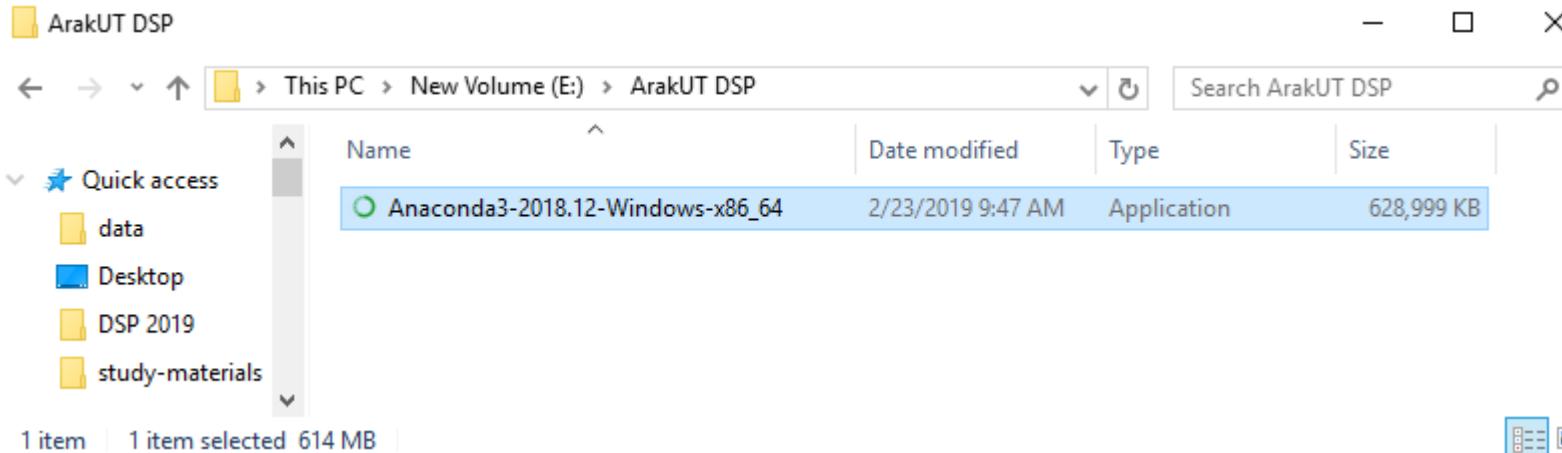
[Download](#)

[64-Bit Graphical Installer \(560.6 MB\)](#)

[32-Bit Graphical Installer \(458.6 MB\)](#)

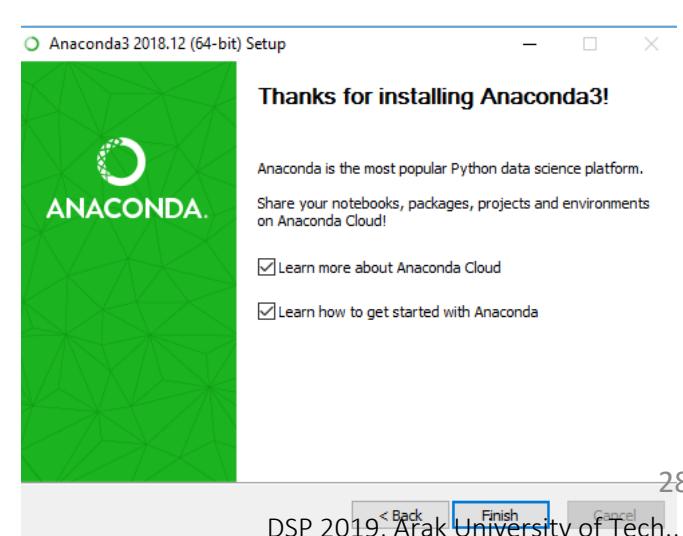
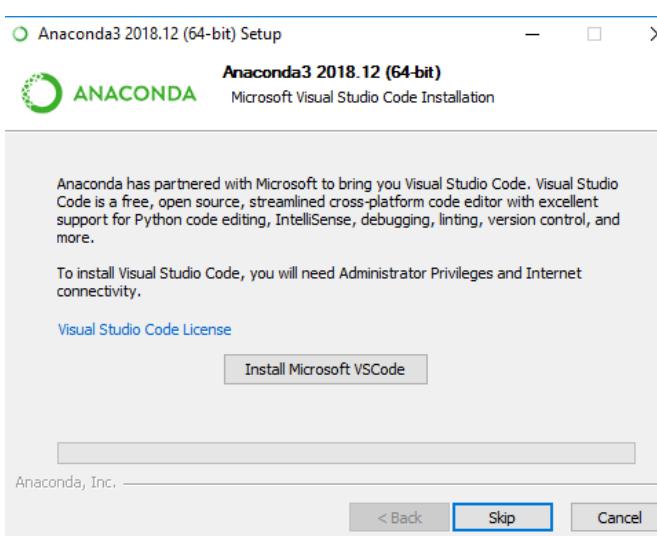
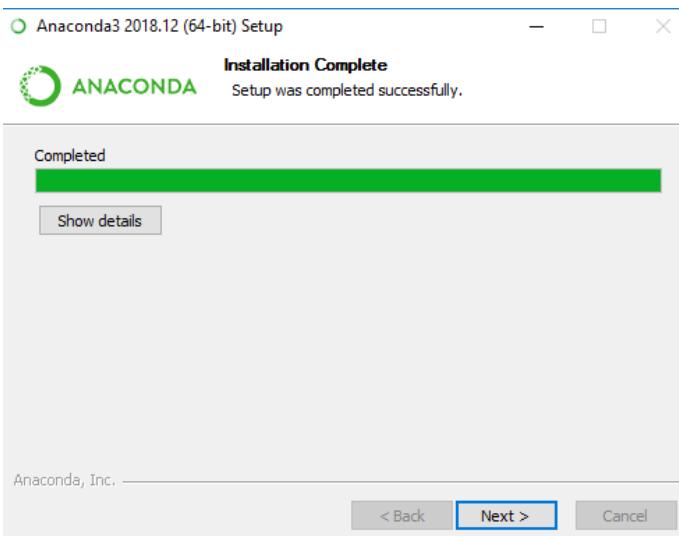
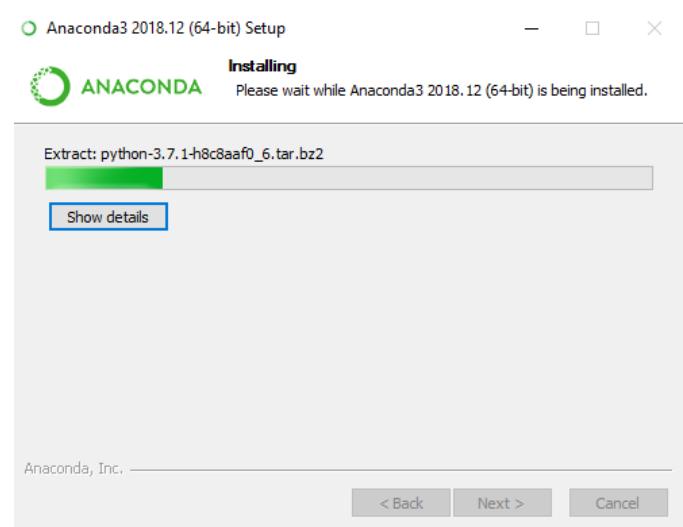
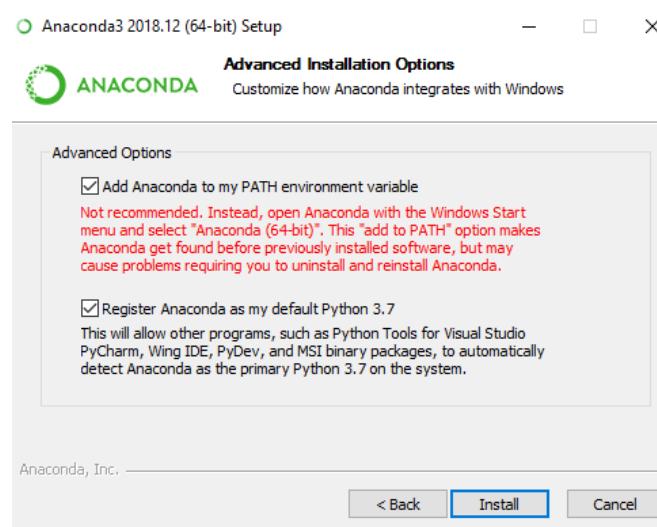
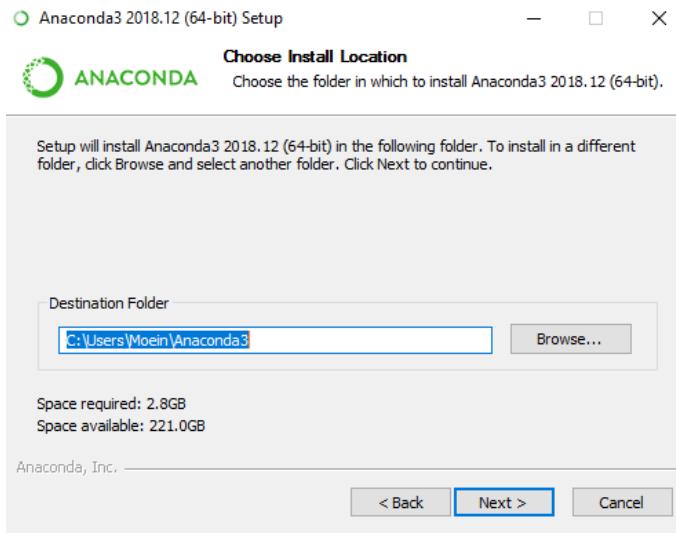


## Getting Started With Python Using Anaconda



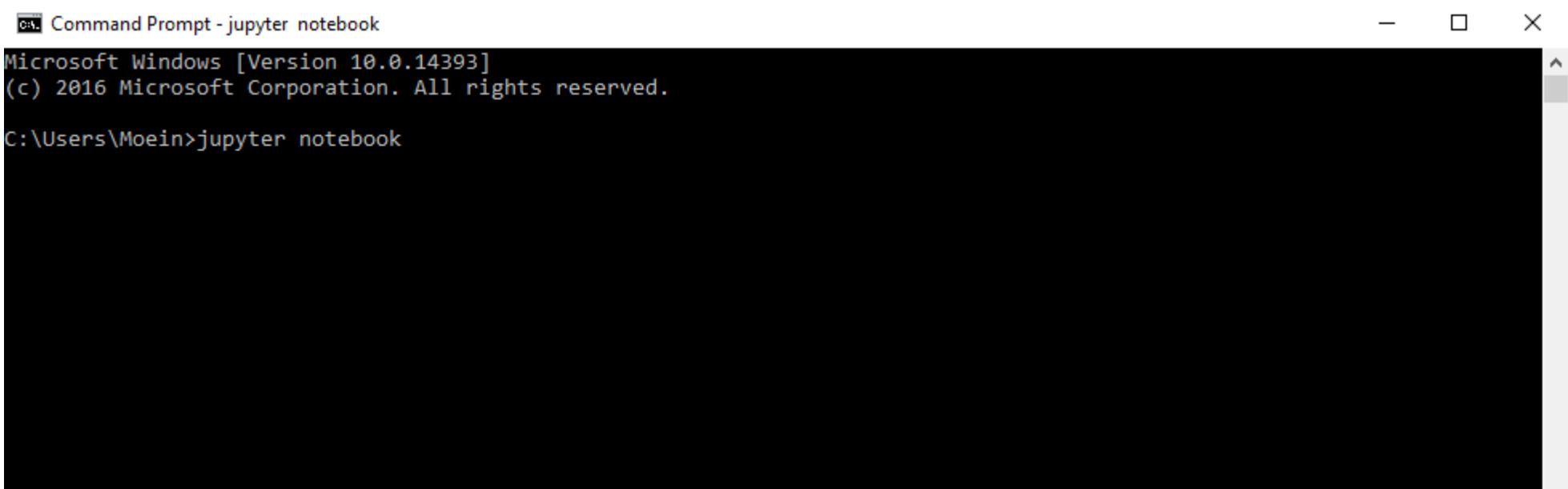


## Getting Started With Python Using Anaconda





## Getting Started With Python Using Anaconda



Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Moein>jupyter notebook

localhost:8888/tree

jupyter

Files    Running    Clusters

Select items to perform actions on them.

0 /

- 3D Objects
- Anaconda3
- AndroidStudioProjects
- Contacts
- Desktop

Name: Python 3  
Notebook: Python 3  
Other:  
Text File  
Folder  
Terminal

4 days ago



## Getting Started With Python Using Anaconda



Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.  
C:\Users\Moein>jupyter notebook



localhost:8888/tree

jupyter

Files    Running    Clusters

Select items to perform actions on them.

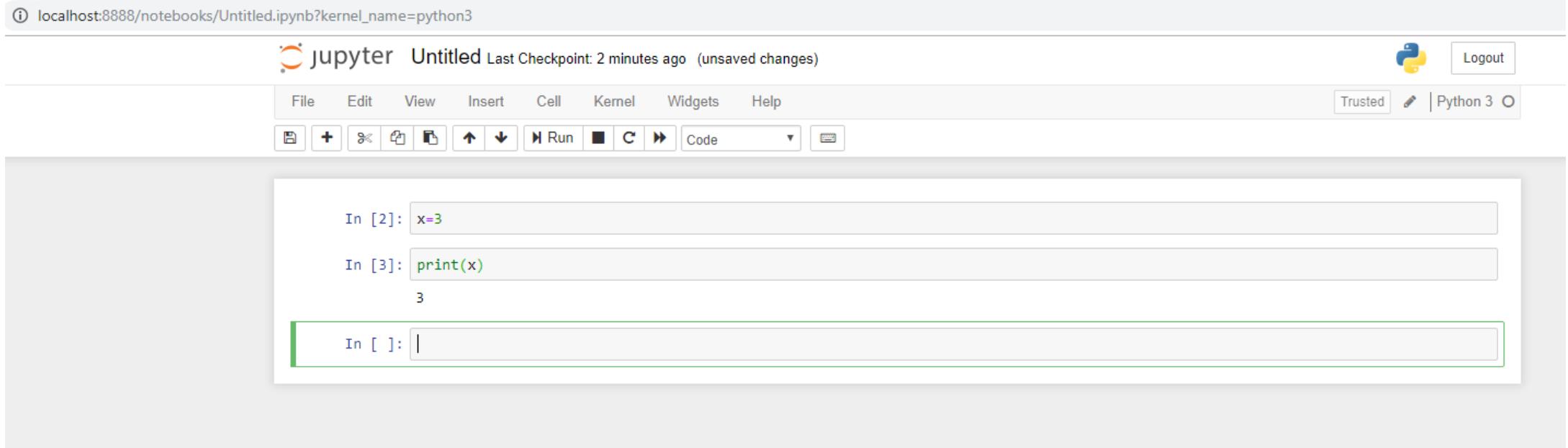
0 /

- 3D Objects
- Anaconda3
- AndroidStudioProjects
- Contacts
- Desktop

Name: Python 3  
Notebook: Python 3  
Other:  
Text File  
Folder  
Terminal

4 days ago

## Getting Started With Python Using Anaconda



The screenshot shows a Jupyter Notebook interface running on localhost:8888. The title bar indicates it's an Untitled notebook using a Python 3 kernel, with a last checkpoint 2 minutes ago and unsaved changes. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and New, as well as Run, Cell, and Kernel controls. The main workspace displays two code cells:

```
In [2]: x=3
In [3]: print(x)
3
```

The cell In [ ]: is currently active, indicated by a green border.

Run: Shift+Enter



## Getting Started With Python Using Anaconda

Anaconda Navigator

File Help

### ANA CONDA NAVIGATOR

Sign in to Anaconda Cloud

Applications on base (root) Channels Refresh

Icon	Name	Version	Description	Action
	JupyterLab	0.35.3	An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.	<a href="#">Launch</a>
	Jupyter Notebook	5.7.4	Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.	<a href="#">Launch</a>
	IP[y]: Qt Console	4.4.3	PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.	<a href="#">Launch</a>
	Spyder	3.3.2	Scientific Python Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.	<a href="#">Launch</a>
	VS Code	1.28.2	Streamlined code editor with support for development operations like debugging, task running and version control.	<a href="#">Launch</a>
	Glueviz	0.13.3	Multidimensional data visualization across files. Explore relationships within and among related datasets.	<a href="#">Install</a>
	Orange 3	3.17.0	Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows	<a href="#">Launch</a>
	RStudio	1.1.456	A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.	<a href="#">Launch</a>

Home Environments Learning Community Documentation Developer Blog

[Twitter](#) [YouTube](#) [GitHub](#)

Anaconda Navigator

File Help

### ANA CONDA NAVIGATOR

Sign in to Anaconda Cloud

Search Environments base (root)

Installed Channels Update index... num

Name	Description	Version
blaze	Numpy and pandas interface to big data	0.11.3
bottleneck	Fast numpy array functions written in cython.	1.2.1
mkl_fft	Numpy-based implementation of fast Fourier transform using intel (r) math kernel library.	1.0.6
mkl_random	Intel (r) mkl-powered package for sampling from various distributions.	1.0.2
numba	Numpy aware dynamic python compiler using llvm.	0.41.0
numexpr	Fast numerical expression evaluator for numpy.	2.6.8
numpy	Array processing for numbers, strings, records, and objects.	1.15.4
numpy-base		1.15.4
numpydoc	Sphinx extension to support docstrings in numpy format.	0.8.0
pytables	Brings together python, hdf5 and numpy to easily handle large amounts of data.	3.4.4
wcwidth	Measures number of terminal column cells of wide-character codes.	0.1.7

Documentation Developer Blog

[Create](#) [Clone](#) [Import](#) [Remove](#)

11 packages available matching "num"

32

DSP 2019, Arak University of Tech., Moein Ahmadi



## 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)  
spamsplamsplams
```

```
>>> 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
```

```
>>> this_is_a_normal_name = 7  
  
>>> 123abc = 7  
SyntaxError: invalid syntax  
  
>>> spaces are not allowed  
SyntaxError: invalid syntax
```

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

```
>>> 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**

```
>>> not 1 == 1
```

**False**

```
>>> not 1 > 7
```

**True**

```
>>> 1 == 1 or 2 == 2
```

**True**

```
>>> 1 == 1 or 2 == 3
```

**True**

```
>>> 1 != 1 or 2 == 2
```

**True**

```
>>> 2 < 1 or 3 > 6
```

**False**

## Operator Precedence

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

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



## while Loops

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

```
>>>  
1  
2  
3  
4  
5  
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 function(variable):
    variable += 1
    print(variable)
```

```
function(7)
print(variable)
```

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

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

```
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()
```

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

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



## 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']"
```

## object-oriented programming (OOP)

### Classes

```
class Cat:  
    def __init__(self, color, legs):  
        self.color = color  
        self.legs = legs
```

```
felix = Cat("ginger", 4)  
rover = Cat("dog-colored", 4)  
stumpy = Cat("brown", 3)
```

```
class Dog:  
    def __init__(self, name, color):  
        self.name = name  
        self.color = color  
  
    def bark(self):  
        print("Woof!")  
  
fido = Dog("Fido", "brown")  
print(fido.name)  
fido.bark()
```

## object-oriented programming (OOP)

### Classes

```
class Cat:  
    def __init__(self, color, legs):  
        self.color = color  
        self.legs = legs
```

```
felix = Cat("ginger", 4)  
rover = Cat("dog-colored", 4)  
stumpy = Cat("brown", 3)
```

```
class Dog:  
    def __init__(self, name, color):  
        self.name = name  
        self.color = color  
  
    def bark(self):  
        print("Woof!")  
  
fido = Dog("Fido", "brown")  
print(fido.name)  
fido.bark()
```

## Classes

### Inheritance

```
class Animal:  
    def __init__(self, name, color):  
        self.name = name  
        self.color = color  
  
class Cat(Animal):  
    def purr(self):  
        print("Purr...")  
  
class Dog(Animal):  
    def bark(self):  
        print("Woof!")  
  
fido = Dog("Fido", "brown")  
print(fido.color)  
fido.bark()
```

```
class A:  
    def spam(self):  
        print(1)  
  
class B(A):  
    def spam(self):  
        print(2)  
        super().spam()  
  
B().spam()
```

# 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  $\Sigma$

Shift-invariant

LSI

## Linearity

$$\begin{aligned} \text{If } x_1(n) &\longrightarrow y_1(n) \\ \text{& } x_2(n) &\longrightarrow y_2(n) \end{aligned}$$

$$\text{then: } \overbrace{ax_1(n) + bx_2(n)}^{\rightarrow ay_1(n) + by_2(n)}$$

$$\sum a_k x_k(n) \longrightarrow$$

$$\sum a_k y_k(n)$$

## Shift-invariance

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

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

(unit sample response)

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

# Class Review

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

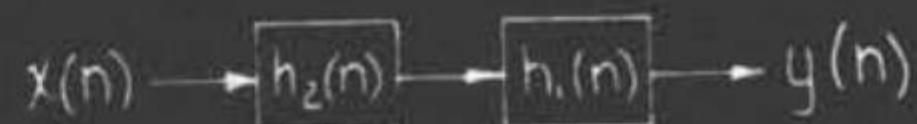
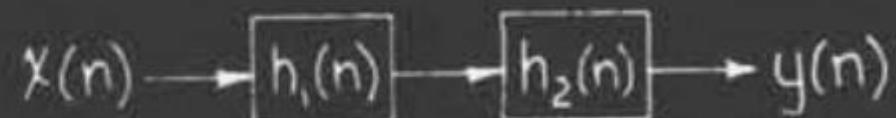
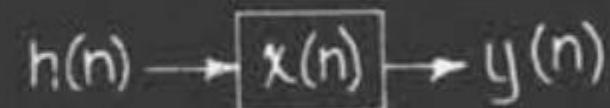
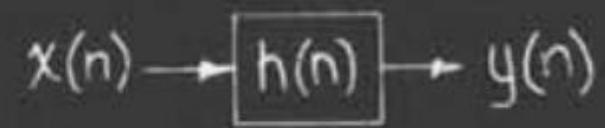
$$y(n) = \sum_{k=-\infty}^{+\infty} x(k) h(n-k)$$

Convolution sum

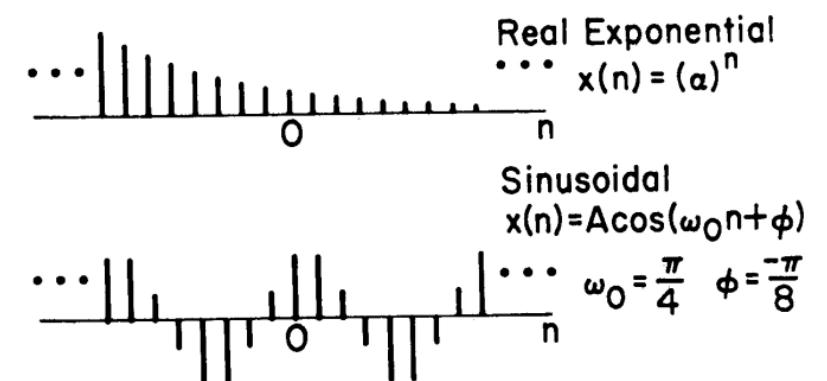
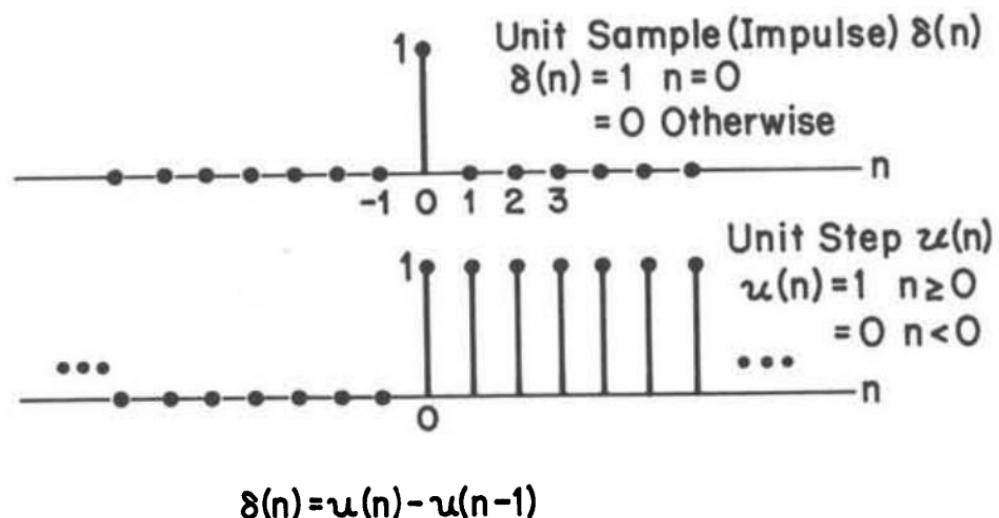
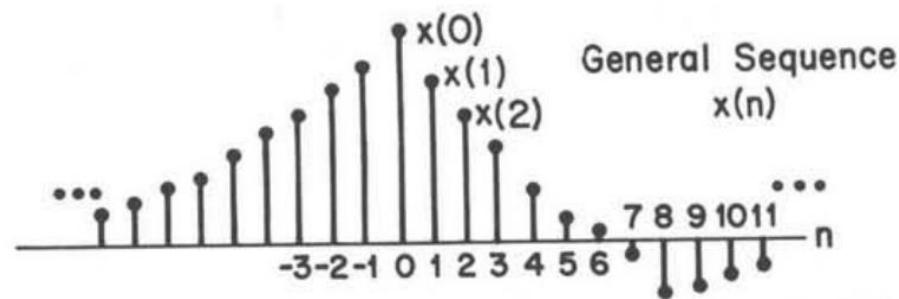
$$n-k=r; k=n-r$$

$$y(n) = \sum_r x(n-r) h(r)$$

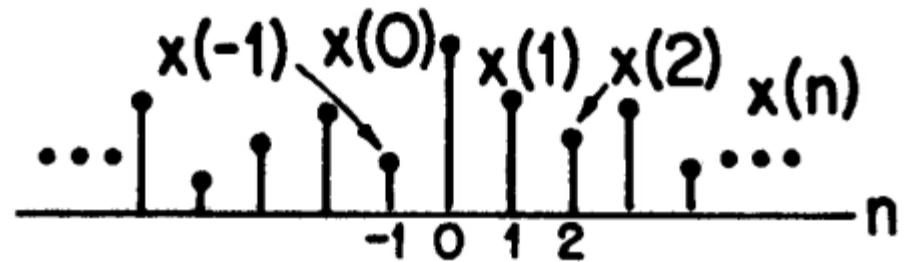
$$\begin{aligned} y(n) &= x(n) * h(n) \\ &= h(n) * x(n) \end{aligned}$$



# Class Review



# Class Review



$$\begin{aligned}
 x(n) &= \\
 &x(0)\delta(n) + x(1)\delta(n-1) \\
 &+ x(-1)\delta(n+1) + \dots \\
 &= \sum_{k=-\infty}^{+\infty} x(k)\delta(n-k)
 \end{aligned}$$

# Class Review

$$x(n) = -T[\cdot] = -y(n)$$

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

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

$$h(n) = 2^n u(n) \text{--- unstable}$$

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

## Causality

$y(n)$  for  $n=n_0$  depends  
on  $x(n)$  only for  $n \leq n_0$

LSI

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

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

noncausal  
stable

# Class Review

$$x(n) = -T[\cdot] = -y(n)$$

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

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

$$h(n) = 2^n u(n) \text{--- unstable}$$

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

## Causality

$y(n)$  for  $n=n_0$  depends  
on  $x(n)$  only for  $n \leq n_0$

LSI

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

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

noncausal  
stable

# Class Review

Linear Constant Coefficient  
Difference Equation

$N^{\text{th}}$  order

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

$$N=0 \quad a_0=1$$

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

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

$$N \neq 0 \quad a_0=1$$

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

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

$$\text{assume } y(n)=0 \quad n>0$$

$$y(n-1) = a' [y(n) - \delta(n)]$$

First-order

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

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

$$\text{assume } y(n)=0 \quad n<0$$

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

$$\left. \begin{array}{l} y(1)=0 \\ y(0)=0 \\ y(-1)=-a^{-1} \\ y(-2)=-a^{-2} \end{array} \right\} \begin{array}{l} -a^n u(-n-1) \\ |a|<1 \\ \text{unstable} \end{array}$$

$$\left. \begin{array}{l} y(-1)=0 \\ y(0)=1 \\ y(1)=a \\ y(2)=a^2 \end{array} \right\} \begin{array}{l} a^n u(n) \\ |a|<1 \\ \text{stable} \end{array}$$

# Class Review

Frequency Response  
of LSI systems

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

$$\text{Let } x(n) = e^{j\omega n}$$

$$y(n) = \sum_{k=-\infty}^{+\infty} h(k) \underbrace{e^{j\omega(n-k)}}_{(e^{j\omega n}) e^{-jk\omega}}$$

$$y(n) = e^{j\omega n} \underbrace{\sum_{k=-\infty}^{+\infty} h(k) e^{-jk\omega}}_{H(e^{j\omega})}$$

$$y(n) = H(e^{j\omega}) e^{j\omega n}$$

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

$\triangleq$  Frequency Response

Sinusoidal Response

$$\begin{aligned} x(n) &= A \cos(\omega_0 n + \phi) \\ &= \frac{A}{2} e^{j\phi} e^{j\omega_0 n} + \frac{A}{2} e^{-j\phi} e^{-j\omega_0 n} \end{aligned}$$

$$H(e^{j\omega_0}) = |H(e^{j\omega_0})| e^{j\theta(\omega_0)}$$

$$\begin{aligned} y(n) &= A |H(e^{j\omega_0})| \\ &\quad \cdot \cos(\omega_0 n + \phi + \theta) \end{aligned}$$

# Class Review

Example

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

causal

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

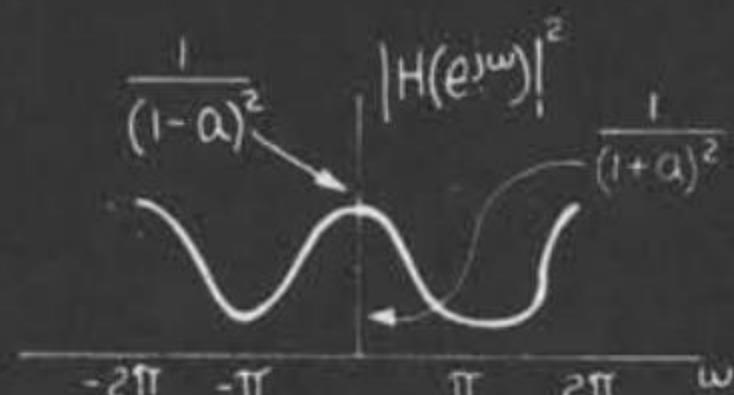
$$H(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} a^n u(n) e^{-jn\omega}$$

$$= \sum_{n=0}^{+\infty} (ae^{-j\omega})^n$$

$$= \frac{1}{1 - ae^{-j\omega}}$$

$$\begin{aligned} |H(e^{j\omega})|^2 &= \frac{1}{1 - ae^{j\omega}} \cdot \frac{1}{1 - ae^{-j\omega}} \\ &= \frac{1}{1 + a^2 - 2a \cos \omega} \end{aligned}$$

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



Properties of Freq Resp

1 Function of continuous variable  $\underline{\omega}$

2 Periodic function of  $\omega$  period  $2\pi$

$$\Rightarrow e^{j(\omega + 2\pi k)n} = e^{j\omega n} e^{j2\pi kn}$$

Generalization  
The Fourier Transform

# Class Review

Frequency Response

$$e^{j\omega n} \longrightarrow H(e^{j\omega}) e^{j\omega n}$$

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

Properties.

1) fcn. of continuous variable  $\omega$

2) periodic - period  $2\pi$

$$H(e^{j\omega}) = H(e^{j(\omega + 2\pi k)})$$

$$h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \underbrace{H(e^{j\omega})}_{\text{Fourier Transform}} e^{j\omega n} d\omega$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[ \sum_k h(k) e^{-j\omega k} \right] e^{j\omega n} d\omega$$

$$= \frac{1}{2\pi} \sum_k h(k) \int_{-\pi}^{\pi} e^{j\omega(n-k)} d\omega$$

$n \neq k$	0
$n = k$	$2\pi$

$$= h(n)$$

Fourier Transform

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

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega$$

$$= \lim_{\Delta\omega \rightarrow 0} \sum_k \left[ X(e^{j\omega_k}) \frac{\Delta\omega}{2\pi} \right] e^{jk\omega_n}$$

Convolution Property

$$x(n) * h(n) \longleftrightarrow X(e^{j\omega}) H(e^{j\omega})$$

# Class Review

→ LSI →

$$e^{j\omega_0 n} \longrightarrow H(e^{j\omega_0}) e^{j\omega_0 n}$$

$$\sum_k A_k e^{j\omega_k n} \rightarrow \sum_k A_k H(e^{j\omega_k}) e^{j\omega_k n}$$

$$\chi(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega$$

$$\underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) H(e^{j\omega}) e^{j\omega n} d\omega}_{=y(n)}$$

$$\Psi(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$$

Symmetry Properties  
 $\chi(n)$  real

$$X(e^{j\omega}) = X^*(e^{-j\omega})$$

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

$$X^*(e^{-j\omega}) = \sum_{n=-\infty}^{+\infty} \underbrace{\chi(n)}_{\bar{\chi}(n)} e^{\bar{\Theta} j\omega n}$$

$$X(e^{j\omega}) = X_R(e^{j\omega}) + j X_I(e^{j\omega})$$

$$X^*(e^{-j\omega}) = X_R(e^{-j\omega}) - j X_I(e^{-j\omega})$$

$$X_R(e^{j\omega}) = X_R(e^{-j\omega}) \text{ even}$$

$$X_I(e^{j\omega}) = -X_I(e^{-j\omega}) \text{ odd}$$

$$|X(e^{j\omega})| \text{ even}$$

$$\not X(e^{j\omega}) \text{ odd}$$

# Class Review

$$\chi_A(t) \boxed{\times T} \tilde{\chi}_A(t)$$

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

$$\tilde{\chi}_A(t) \rightarrow \boxed{C|D} \xrightarrow{\chi(n)} \chi_A(nT)$$

$$\begin{aligned}\tilde{\chi}_A(t) &= \chi_A(t) \sum_{n=-\infty}^{+\infty} u_o(t-nT) \\ &= \sum_{n=-\infty}^{+\infty} \chi_A(nT) u_o(t-nT)\end{aligned}$$

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

$$\begin{aligned}\tilde{X}_A(j\Omega) &= \frac{1}{T} \sum_{r=-\infty}^{+\infty} X_A(j\Omega + \frac{2\pi r}{T}) \\ &= \sum_{n=-\infty}^{+\infty} \chi_A(nT) e^{jn\Omega T}\end{aligned}$$

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

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

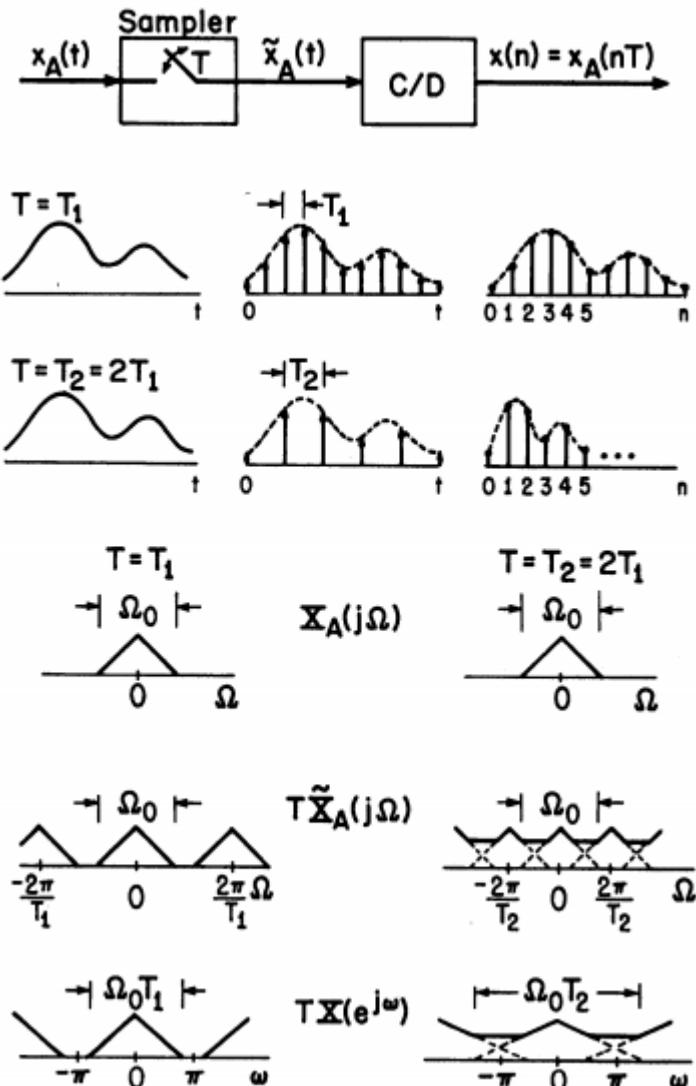
$$X(e^{j\omega}) = \sum_n \chi_A(nT) e^{-j\omega n}$$

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

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

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

# Class Review



# Class Review

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

$$\begin{aligned}|X(e^{j\omega})| &= \left| \sum_{-\infty}^{+\infty} x(n) e^{-jn\omega} \right| \\ &\leq \sum_{-\infty}^{+\infty} |x(n)| |e^{-jn\omega}| \end{aligned}$$

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

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

stable system

$\curvearrowleft H(e^{j\omega})$  converges

Example

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

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

$$\textcircled{2} \quad x(n) = (2)^n u(n)$$

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

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

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

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

Example

$$x(n) = (1/2)^n u(n)$$

$$X(z) = \sum_{n=0}^{\infty} (1/2)^n z^n = \sum_{n=0}^{\infty} (\frac{1}{2}z)^n$$

$$\frac{1}{2}z^n = \frac{z}{2} z^n$$

$$\therefore |(\frac{1}{2}z)^n| < \infty \Rightarrow |z| > \frac{1}{2}$$

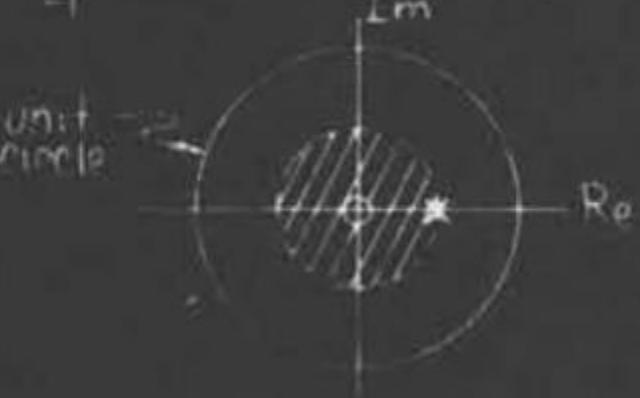


Example

$$y(n) = (1/2)^n u(n-1)$$

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

$$\sum_{n=1}^{\infty} \left| \left( \frac{1}{2}z^{-1} \right)^n \right| < \infty \Rightarrow |z| < \frac{1}{2}$$



(1) Region of Convergence  
bounded by poles or (0/ $\infty$ )

(2) Finite Length Sequences

$$0 < |z| < \infty$$

(3) Right-sided  $x(n) = 0 \quad n < n_0$ ,  
 $R_{x-} < |z| < \infty$

$$0 < |z| < R_{x+}$$

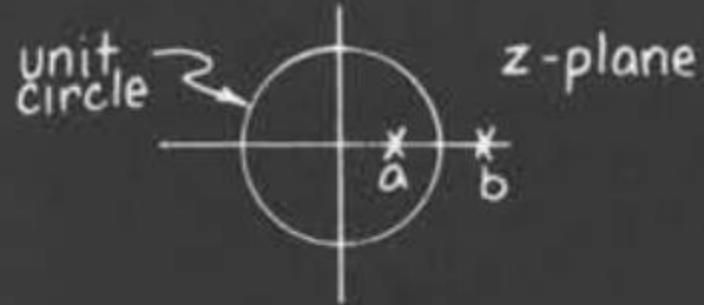
(4) Left-sided  $x(n) = 0 \quad n > n_0$ ,

$$0 < |z| < R_{x-}$$

(5) Two-sided

$$R_{x-} < |z| < R_{x+}$$

# Class Review



$$y(n) = \chi(n) * h(n)$$

$$\Upsilon(z) = \sum_{n=-\infty}^{\infty} \chi(n) H(z)$$

Example  
 $y(n) - \frac{1}{2}y(n-1) = \chi(n)$   
 useful property:

Region of Convergence which Fourier Sided Transform?

$|z| < a$  left no

$a < |z| < b$  two yes

$b < |z|$  right no

$H(z) \triangleq$  System Function

✓ Stable  $\longleftrightarrow$  unit circle in R.of C.

causal  $\implies h(n)$  right-sided  
 $\implies$  R.of C. outside outermost pole

$y(n) \longleftrightarrow \Upsilon(z)$

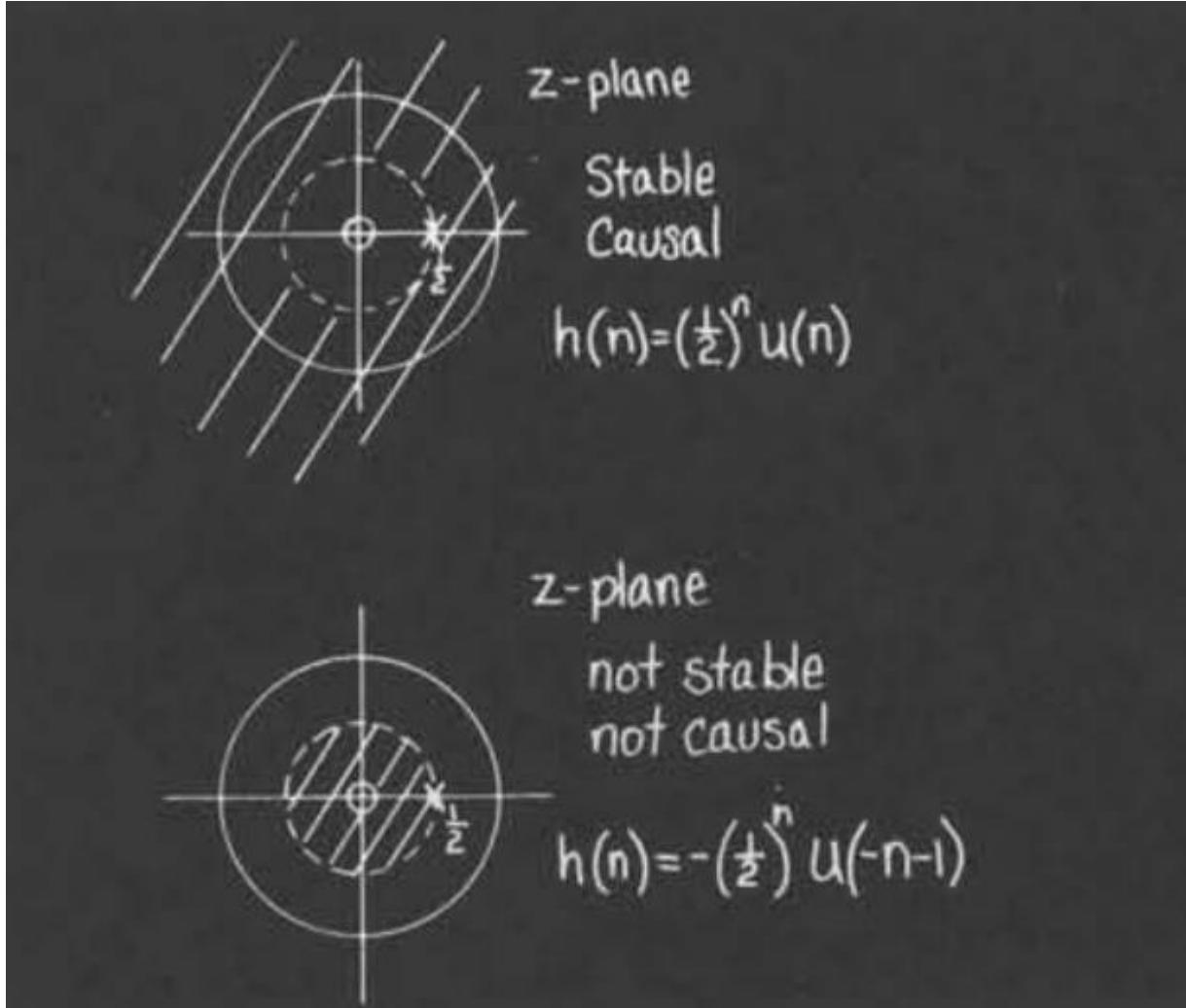
$y(n+n_0) \longleftrightarrow z^{n_0} \Upsilon(z)$

$\Upsilon(z) - \frac{1}{2}z^{-1}\Upsilon(z) = \sum_{n=-\infty}^{\infty} \chi(n) H(z)$

$$H(z) = \frac{\Upsilon(z)}{\sum_{n=-\infty}^{\infty} \chi(n) H(z)}$$

$$= \frac{1}{1 - \frac{1}{2}z^{-1}}$$

# Class Review



# Class Review

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

Inverse z-transform

Inspection Method

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

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

ETC.

## 2. Power Series

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

$$|z| > |a|$$

$$\frac{1+a z^{-1}+a^2 z^{-2}+a^3 z^{-3}+\dots}{1-a z^{-1}}$$

$$|z| < |a|$$

but also

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

## 3. Partial Fraction Exp.

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

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

$$= \sum_{k=1}^N \underbrace{\frac{R_k}{(x-x_k)} (x-x_r)}_{\begin{cases} = 0 & k \neq r \\ = R_r & k = r \end{cases}} \Big|_{(x=x_r)}$$

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

= Residue of  $F(z)$  at  $x_r$

# Class Review

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

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

Example

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

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

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

$$\frac{8}{(z^{-1}-2)(z^{-1}-4)} \Big|_{z^{-1}=2} = -4 \quad 4. \text{ Contour Integration}$$

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

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

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

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

$$= \underbrace{\frac{2}{1 - \frac{1}{2}z^{-1}}}_{z(\frac{1}{2})^n u(n)} + \underbrace{\frac{-1}{1 - \frac{1}{4}z^{-1}}}_{z(\frac{1}{4})^n u(n)}$$

Example

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

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

# Class Review

unit circle



$z$ -plane  $n \geq 0$

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

$$\chi(n) = \frac{-1}{2\pi j} \oint_{C'} X(\gamma_p) p^{-n-1} dp$$

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

$n$  poles at  $z = 0$

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

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

Easy way:

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

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

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

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

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

If  $z = r e^{j\theta}$   
then  $p = (|r|) e^{-j\theta}$



# Class Review

Previous Example

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



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

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

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

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

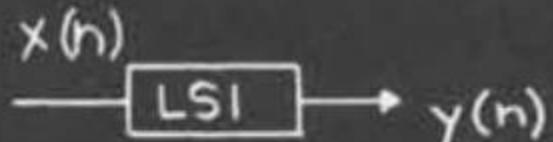
$$\therefore X(n) = 0 \quad n < 0$$

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

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

$$X(n) = (\frac{1}{2})^n \delta(n)$$

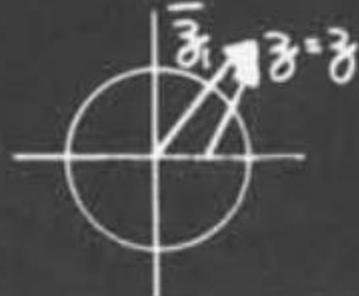
# Class Review



$$\Sigma(\bar{z}) H(\bar{z}) = Y(\bar{z})$$

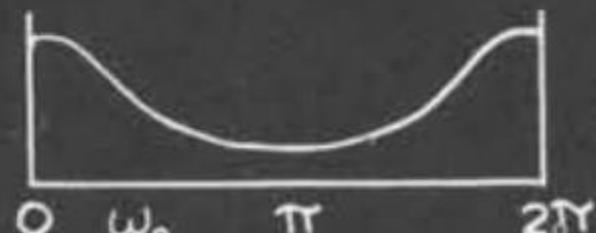
$$H(\bar{z}) \Big|_{\bar{z}=e^{j\omega}} = H(e^{j\omega})$$

$$H(\bar{z}) = \frac{1}{1 - a\bar{z}^{-1}} = \frac{\bar{z}}{\bar{z} - a}$$



$$|H(\bar{z})| = |\bar{z}| / |(\bar{z} - a)|$$

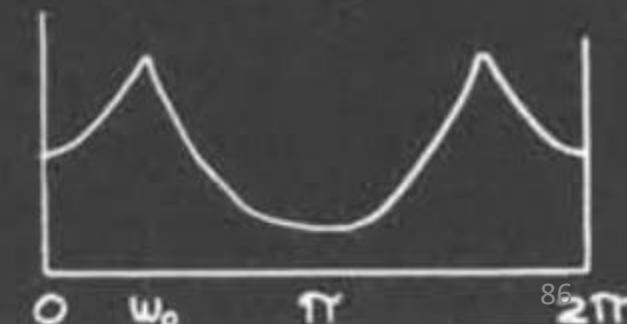
$$H(\bar{z}) = \bar{z} - (\bar{z} - a)$$



General

$$|H(e^{j\omega})| = \frac{\text{length zero}}{\text{length pole}}$$

$$H = \sum \text{zero} - \sum \text{pole}$$



# Class Review

Transform Properties

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

$$1) x(n)*h(n) \longleftrightarrow X(z)H(z)$$

$$2) x(n+n_0) \longleftrightarrow z^{n_0} X(z)$$

$$3) x(-n) \longleftrightarrow X(\frac{1}{z})$$

$$4) a^n x(n) \longleftrightarrow X(a^{-1}z)$$

$$5) n x(n) \longleftrightarrow -z \frac{dX(z)}{dz}$$

⋮

⋮

Property 2

$$x_1(n) = x(n+n_0)$$

$$X_1(z) = \sum_{n=-\infty}^{+\infty} x(n+n_0) z^{-n}$$

$$n+n_0=m \quad n=m-n_0$$

$$X_1(z) = \sum_{m=-\infty}^{+\infty} x(m) \cancel{z^{n_0}} z^{-m}$$

LCDE

$$\sum_{k=0}^n a_k \cancel{y(n-k)} = \sum_{k=0}^m b_k \cancel{x(n-k)}$$

$$\frac{Y(z)}{X(z)} = \frac{\sum_0^m b_k z^k}{\cancel{\sum_0^n a_k z^{-k}}}$$

Property 4

$$x_1(n) = a^n x(n)$$

$$X_1(z) = \sum_{n=-\infty}^{+\infty} \underbrace{a^n x(n)}_{x(n)(a^{-1}z)^{-n}} z^{-n}$$

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

Consider pole | zero

$$X(z) : (z-z_0)$$

$$X_1(z) : (a^{-1}z - z_0) = a^{-1}(z - az_0)$$

# Class Review

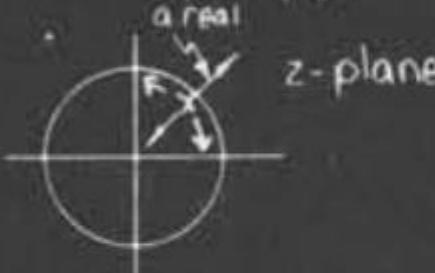
pole(zero)

$z_0$

pole (zero)

$a z_0$

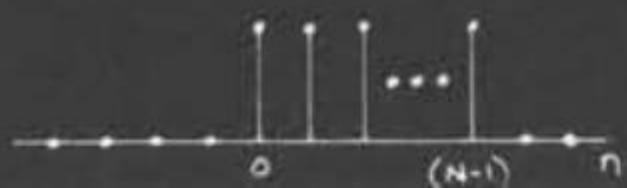
$r_0 e^{j\theta_0}$



Boxcar Sequence

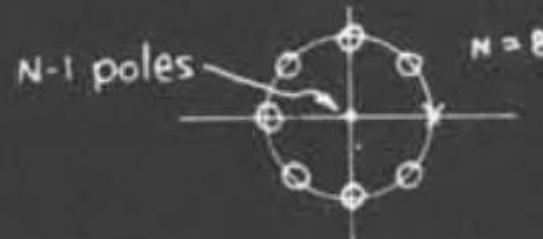
$$\chi(n) = 1 \quad 0 \leq n \leq (N-1)$$

= 0 otherwise



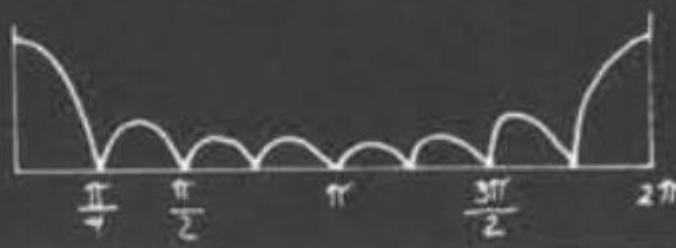
$$\chi(n) = u(n) - u(n-N)$$

$$\begin{aligned} X(z) &= \frac{1}{1-z^{-1}} - \frac{z^{-N}}{1-z^{-1}} \\ &= \frac{1-z^N}{1-z^{-1}} = \frac{z^{N-1}}{z^{(N-1)}(z-1)} \end{aligned}$$



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

$$\begin{aligned} &= \frac{2j \sin \frac{\omega N}{2}}{e^{j\frac{\omega}{2}} [e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}}]} \\ &= \frac{2j \sin \frac{\omega N}{2}}{e^{j\frac{\omega}{2}} [e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}]} \\ &= e^{j\omega \frac{(N-1)}{2}} \frac{\sin(\omega \frac{N}{2})}{\sin(\frac{\omega}{2})} \end{aligned}$$



# Class Review

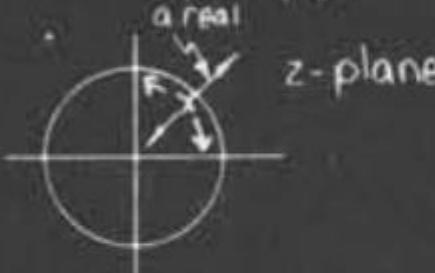
pole(zero)

$z_0$

pole (zero)

$a z_0$

$r_0 e^{j\theta_0}$



$$\chi(n) = u(n) - u(n-N)$$

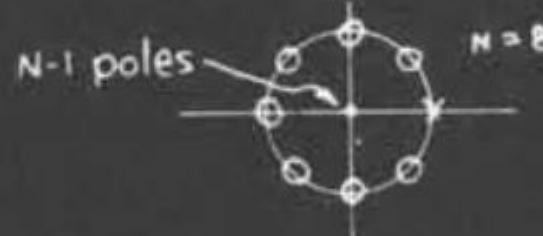
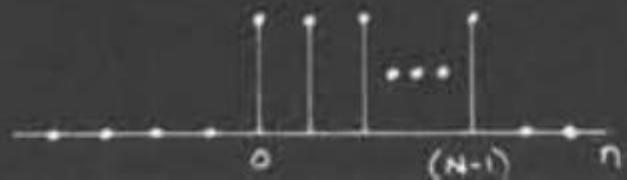
$$X(z) = \frac{1}{1-z^{-1}} - \frac{z^{-N}}{1-z^{-1}}$$

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

Boxcar Sequence

$$\chi(n) = 1 \quad 0 \leq n \leq (N-1)$$

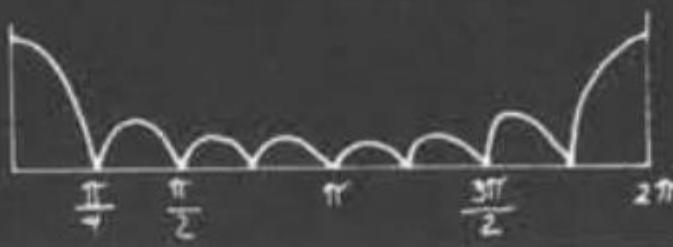
= 0 otherwise



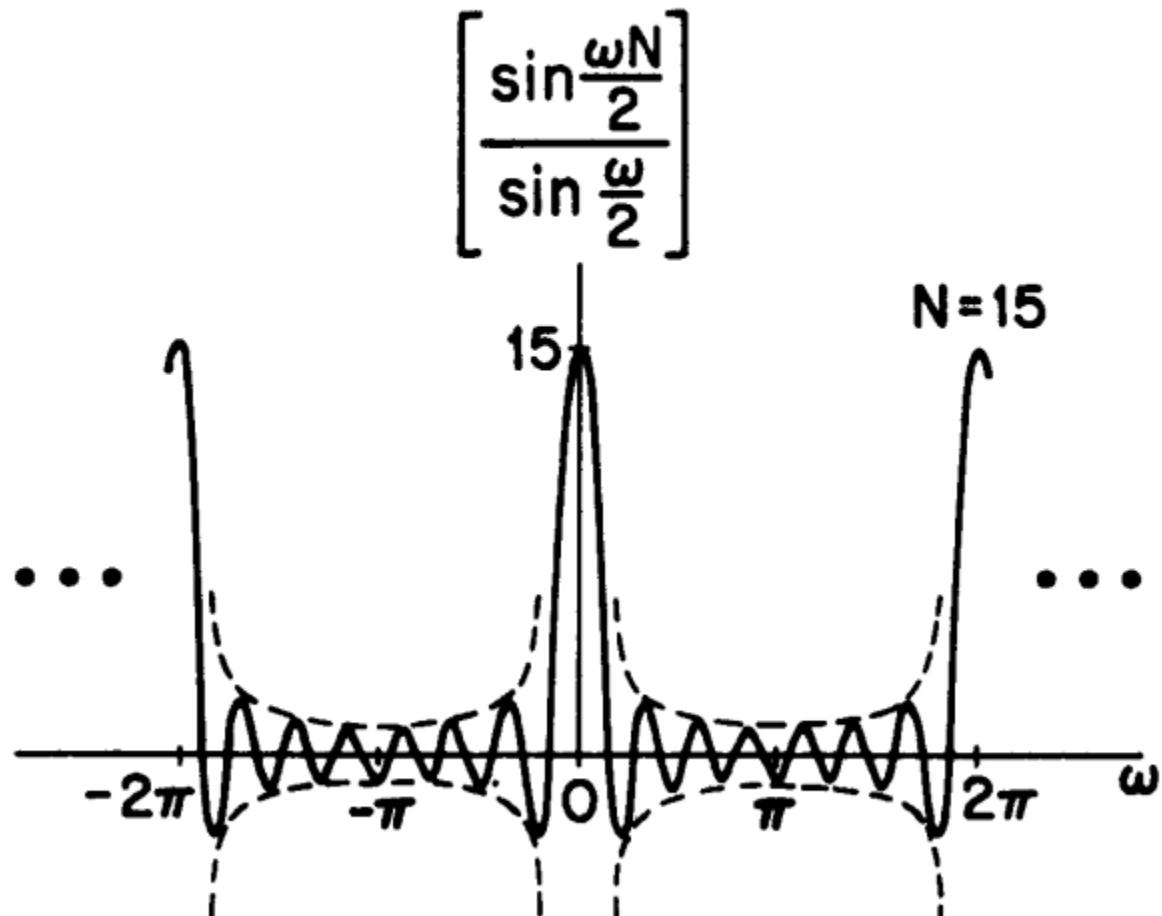
$$X(e^{j\omega}) = \frac{1-e^{j\omega N}}{1-e^{j\omega}}$$

$$= \frac{e^{j\omega N} [e^{j\omega \frac{N}{2}} - e^{-j\omega \frac{N}{2}}]}{e^{j\omega} [e^{j\omega \frac{N}{2}} - e^{-j\omega \frac{N}{2}}]}$$

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



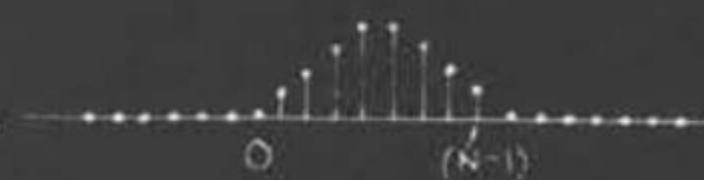
# Class Review



Fourier Transform of a rectangular sequence.

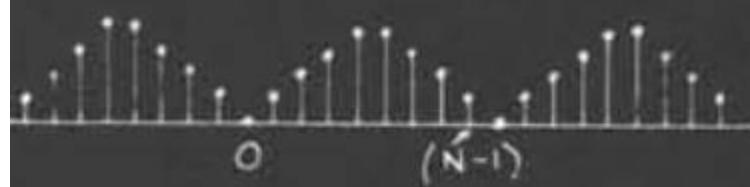
# Class Review

$x(n)$  Finite length



$\tilde{x}(n)$  Periodic

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



$$x(n) = \begin{cases} \tilde{x}(n) & 0 \leq n \leq (N-1) \\ 0 & \text{otherwise} \end{cases}$$

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

$$R_N(n) = \begin{cases} 1 & 0 \leq n \leq (N-1) \\ 0 & \text{otherwise} \end{cases}$$

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

DFT 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

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

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

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

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

$\tilde{X}(k)$  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

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

$$w_N^{ln} \tilde{x}(n)$$

$$\tilde{X}(k+l)$$

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

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

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

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

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

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

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

$$\not|\tilde{X}(k) \text{ odd}$$

# 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

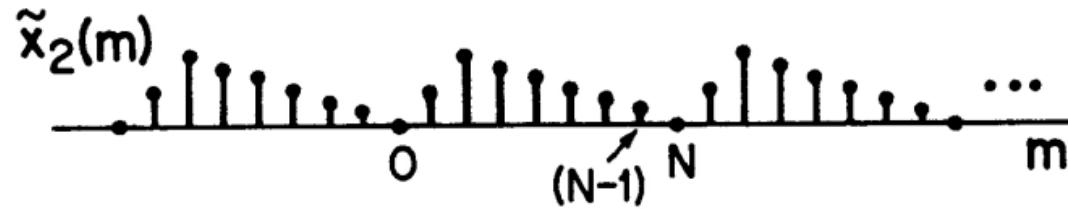
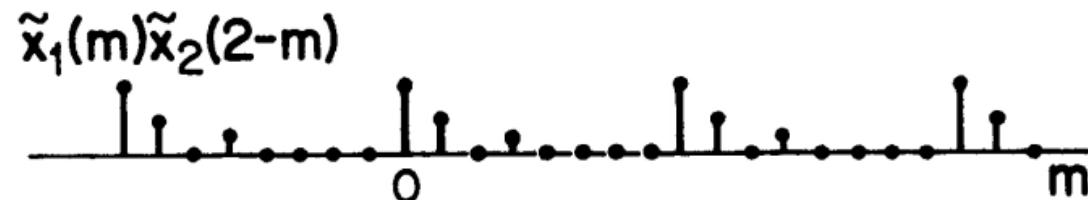
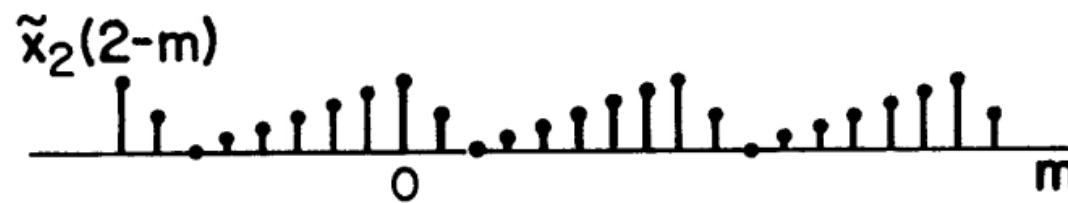


Illustration of the sequences involved in forming a periodic convolution.



# Class Review

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

finite length  $N$   
(or less)

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

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

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

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

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

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

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

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

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

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

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

$$\tilde{X}(k) = \tilde{X}(z) \Big|_{z=W_N^k} \quad N=8$$



# Class Review

## Properties of the DFT

### Shifting Property

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

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

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

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

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

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

## Symmetry Properties

DFT  $x(n)$  real

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

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

DFT  $x(n)$  real

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

(even)

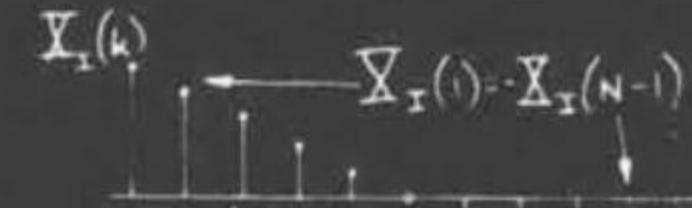
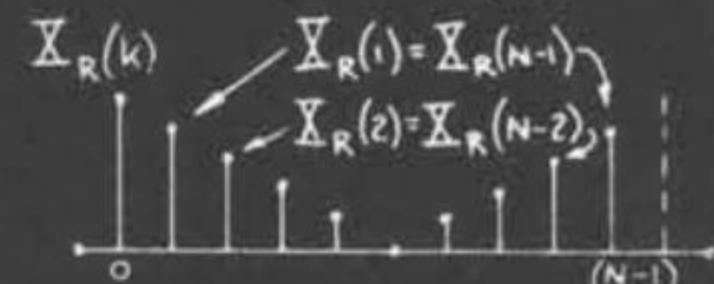
$$X_I(k) = -\tilde{X}_I((N-k))_N R_N(k)$$

(odd)

## for example

$$\tilde{X}_R(0) = \tilde{X}_R((N-0))_N R_N(0)$$

$$\tilde{X}_R(1) = \tilde{X}_R((N-1))_N R_N(1)$$



$$X_I(2) = -\tilde{X}_I(N-2)$$

# Class Review

Convolution Property

$$x_3(n) \longleftrightarrow X_1(k) X_2(k)$$

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

$$x_3(n) = \tilde{x}_3(n) R_N(n)$$

$$x_3(n) = \left[ \sum_{m=0}^{N-1} \tilde{x}_1(m) \tilde{x}_2(n-m) \right] R_N(n)$$

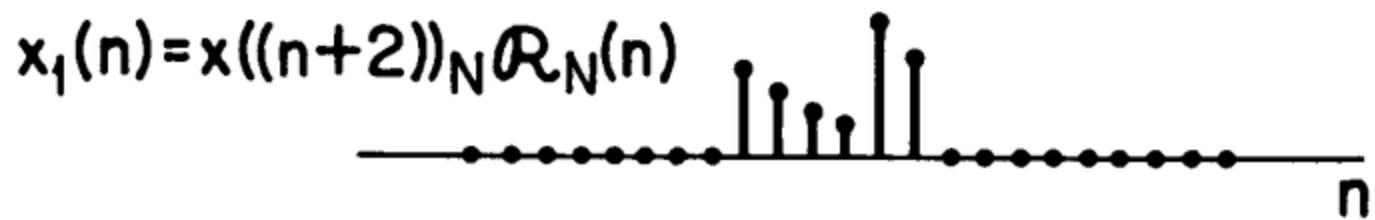
$$= \left[ \sum_{m=0}^{N-1} x_1((m))_N x_2((n-m))_N \right] R_N(n)$$

$$x_3(n) = x_1(n) \circledast x_2(n)$$

# Class Review



Circular shifting of a finite length sequence.



# Class Review

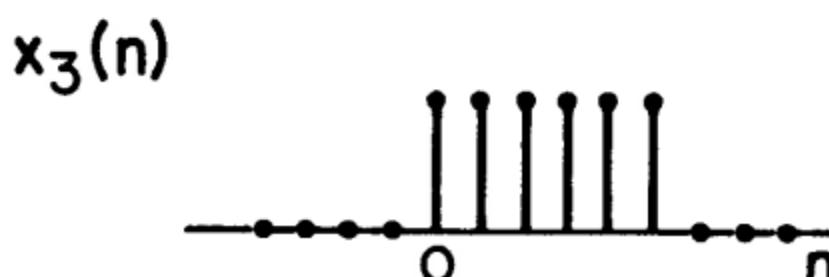
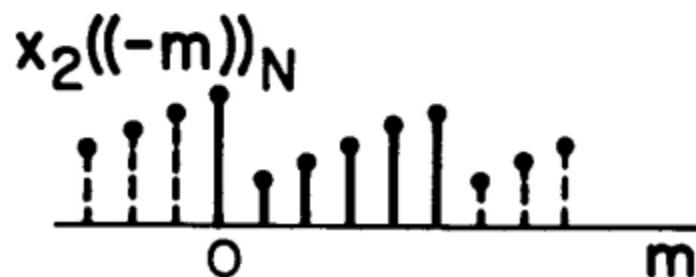


Illustration of circular convolution.  
 (Note that  $x_2((-m))_N$  is incorrectly drawn.  
 In Problem 9.4 you are asked to correct this.)

# Class Review

$$\text{Circular Convolution } x_3(n) = x_1(n) \circledast x_2(n)$$

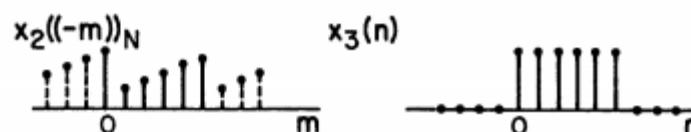
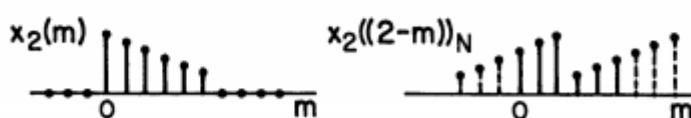
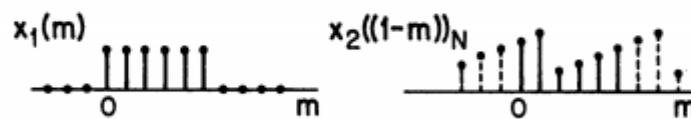
$$= \left[ \sum_{m=0}^{N-1} \tilde{x}_1(m) \tilde{x}_2(n-m) \right] R_N(n)$$

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

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

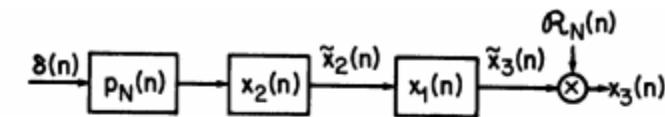
$$= [x_1(n) * x_2((n))]_N R_N(n)$$

a.

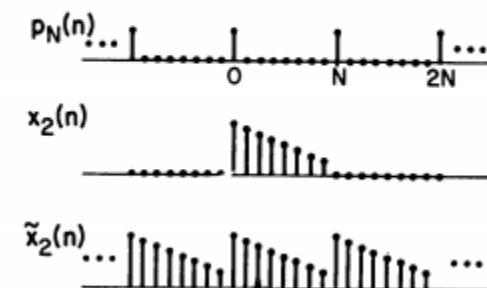


b.

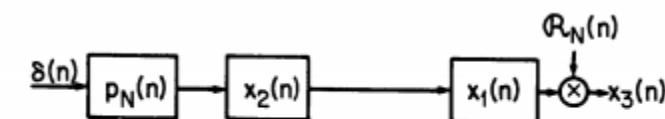
Circular convolution expressed in terms of periodic and linear convolution.



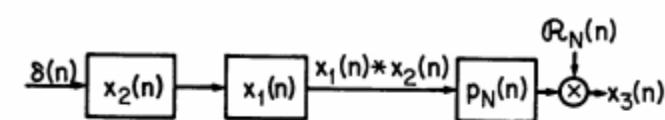
An interpretation of circular convolution.



c.



Rearrangement of the operations in forming the circular convolution.



d.

# Class Review

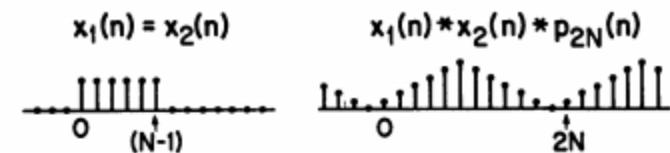
"Circular Convolution =  
Linear Convolution + Aliasing"

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

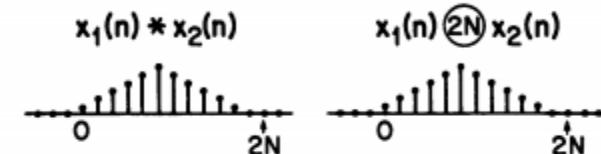
$$x_3(n) = x_1(n) \odot x_2(n)$$

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

Interpretation of  
circular convolution  
as linear convolution  
followed by aliasing.



Obtaining a linear  
convolution through  
the use of circular  
convolution.



e.

$$x_1(n) = x_2(n)$$

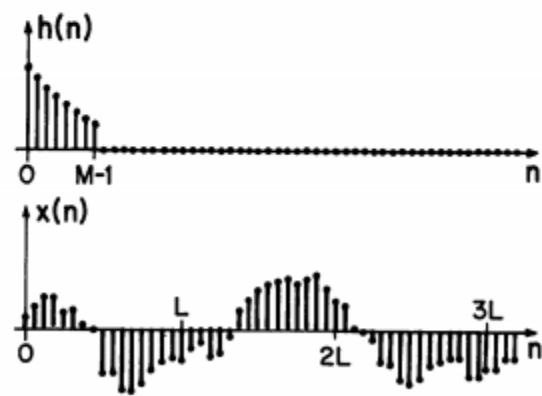


$$x_1(n) * x_2(n) * p_N(n)$$



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

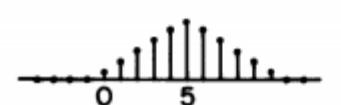
g.



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

f.

$$x_1(n) * x_2(n)$$

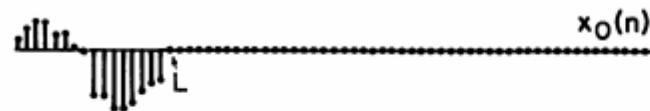


$$x_1(n) \odot x_2(n)$$

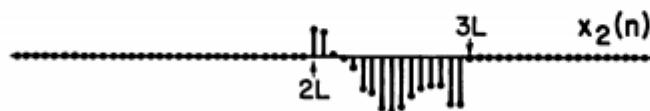
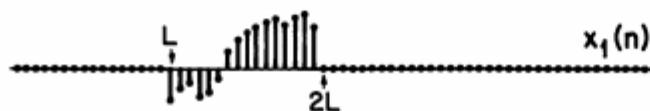


h.

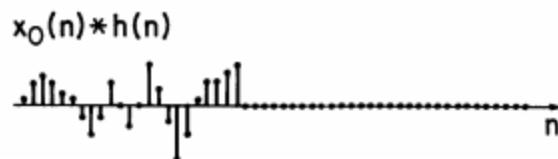
# Class Review



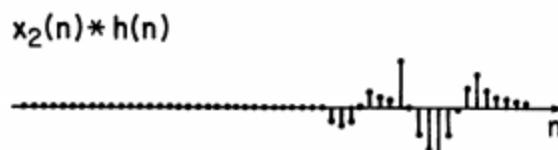
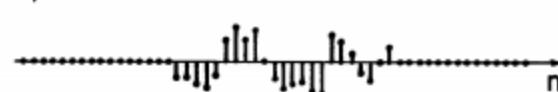
Sectioning of the sequence  $x(n)$ .



i.



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



j.

# Class Review

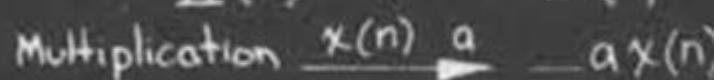
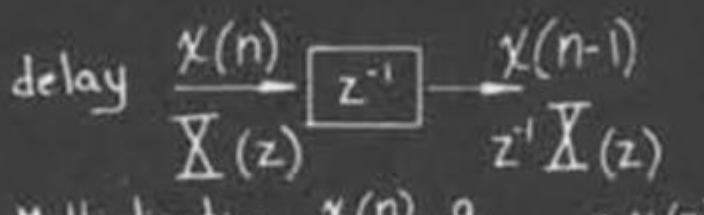
## Digital Networks

$N^{\text{th}}$  order Difference Eq

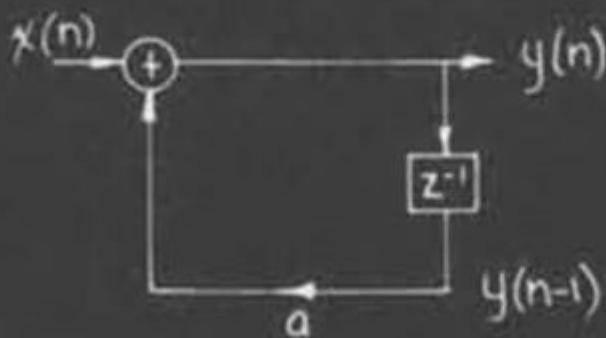
$$y(n) - \sum_{k=1}^N a_k y(n-k) = \sum_{k=0}^M b_k x(n-k)$$

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

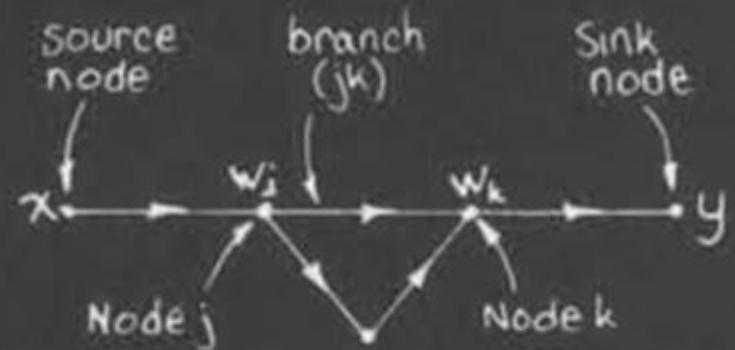
## Basic Operations



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<sup>th</sup> source to k<sup>th</sup> node

# Class Review

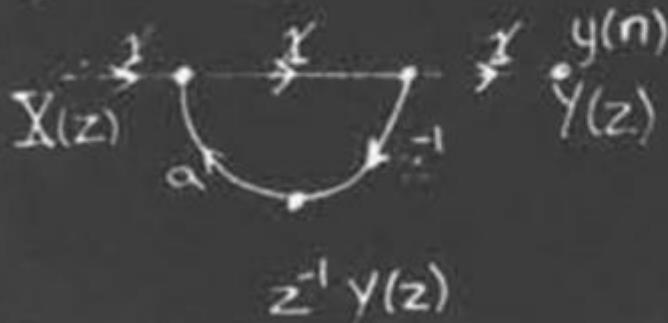
$$W_k = \underbrace{\sum_{j=1}^N V_{jk}}_{\text{Network}} + \underbrace{\sum_{j=1}^M S_{jk}}_{\text{Source}}$$

Linear Signal Flow Graph

arbitrary branch  $(jk)$

$$V_{jk} = t_{jk} W_j$$

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



Matrix Representation  
of Digital Networks



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

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

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

$$W_k(z) = \sum_{j=1}^N F_{jk}(z) W_j(z)$$

$$+ \sum_{j=1}^M b_{jk} X_j(z)$$

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} & \dots & F_{1N} \\ F_{21} & F_{22} & \dots & F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ F_{N1} & F_{N2} & \dots & F_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1N} \\ b_{21} & \dots & b_{2N} \\ \vdots & \ddots & \vdots \\ b_{N1} & \dots & b_{NN} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_M \end{bmatrix}$$

# Class Review

$$\underline{W}(z) = \underline{F}^T(z) \underline{W}(z) + \underline{B}^T \underline{\chi}(z)$$

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

$$\underline{F}^T(z) = F_c^T + z^{-1} F_d^T$$

$$\begin{aligned} W(z) &= F_c^T \underline{W}(z) + z^{-1} F_d^T \underline{W}(z) \\ &\quad + B^T \underline{\chi}(z) \end{aligned}$$

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

or:

$$\underline{W}(n) = F_c^T \underline{W}(n) + F_d^T \underline{W}(n-1) + B^T \underline{\chi}(n)$$

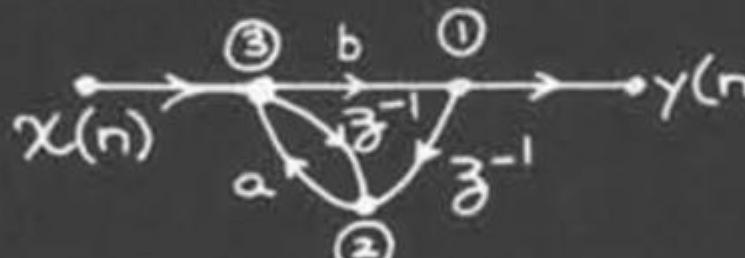
$$y(n) = C^T \underline{W}(n)$$

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

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

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

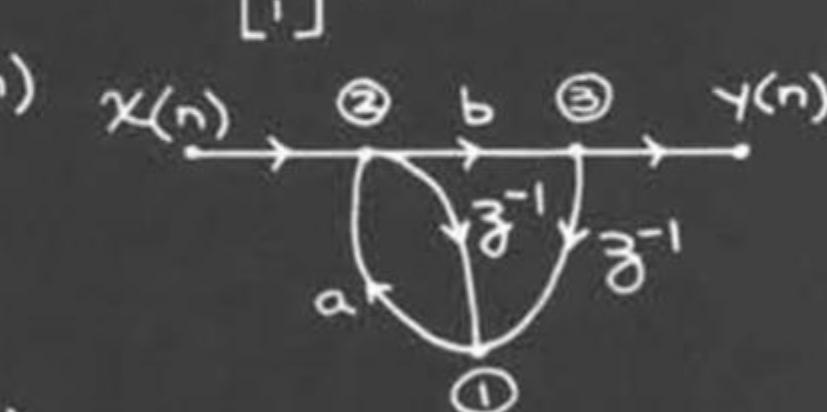
Example



$$W_1(n) = b W_3(n)$$

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

$$W_3(n) = a W_2(n) + \chi(n)$$



# Class Review

$$\begin{bmatrix} \omega_1(n) \\ \omega_2(n) \\ \omega_3(n) \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & b \end{bmatrix} \begin{bmatrix} \omega_1(n) \\ \omega_2(n) \\ \omega_3(n) \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1(n-1) \\ \omega_2(n-1) \\ \omega_3(n-1) \end{bmatrix}$$

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

Network Computable

Nodes can be numbered  
so that  $F_c^t$  is

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



Non computable Network

# Class Review

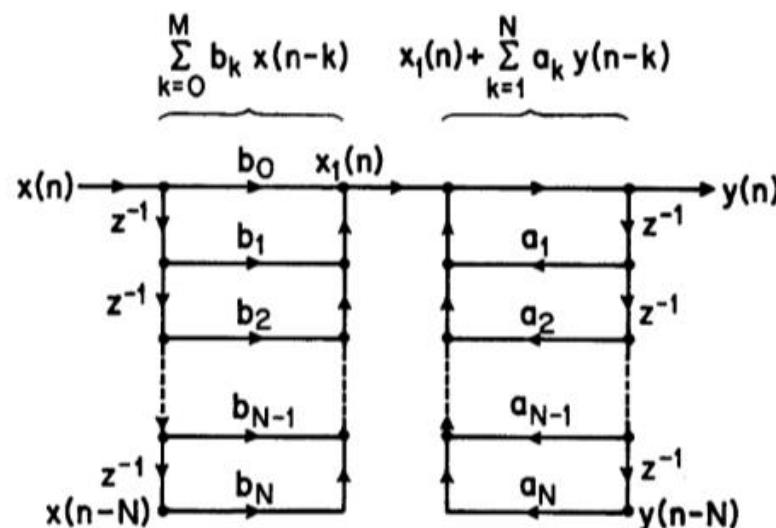
$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 - \sum_{k=1}^N a_k z^{-k}}$$

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

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

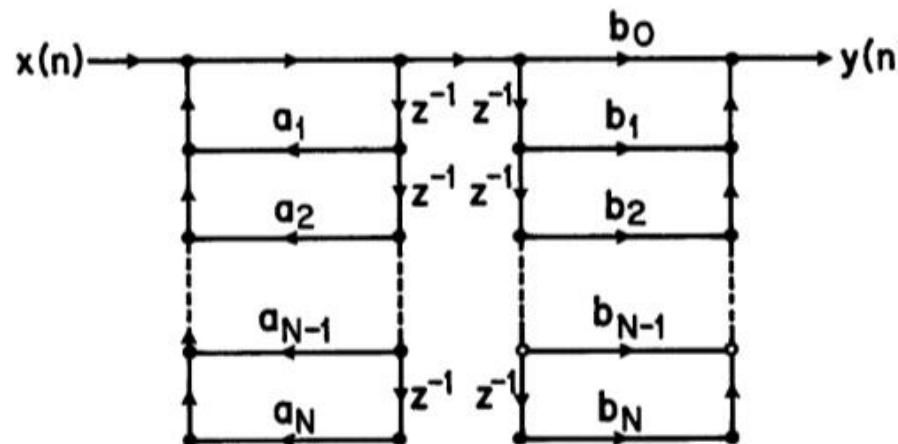
$$H(z) = \left[ \sum_{k=0}^M b_k z^{-k} \right] \left[ \frac{1}{1 - \sum_{k=1}^N a_k z^{-k}} \right]$$

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.

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

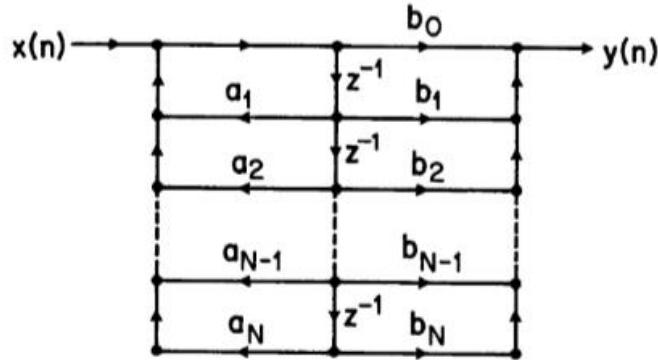
$z$ -transform factorization and difference equation corresponding to the network in c.

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

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

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

# Class Review



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

## TRANSPOSITION THEOREM

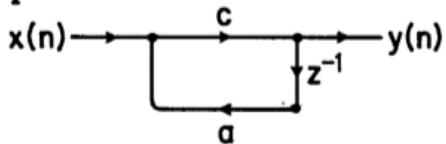
Transposition theorem  
for signal flow-graphs.

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

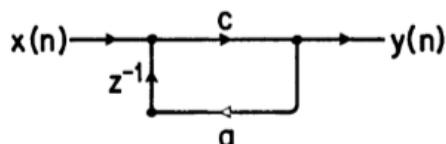
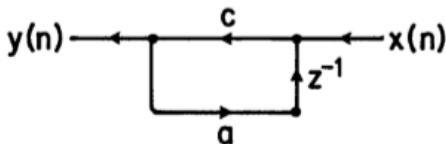
TRANSFER FUNCTION REMAINS THE SAME

# Class Review

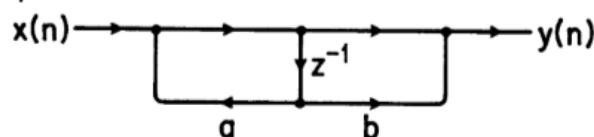
**Example 1**



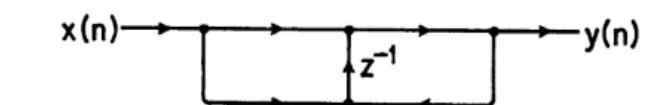
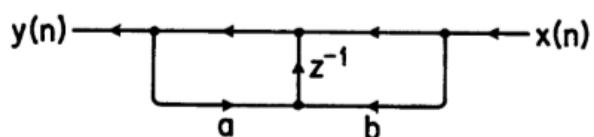
Example of Transposition theorem.



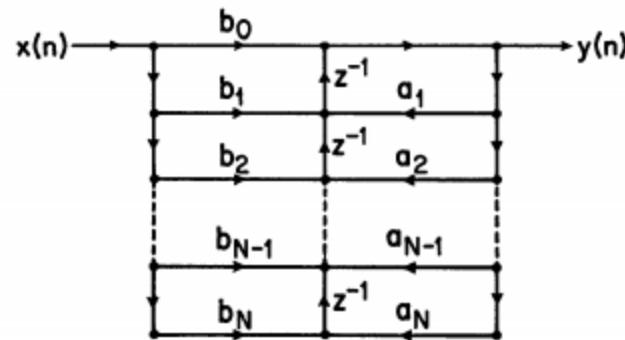
**Example 2**



Example of Transposition theorem.



# Class Review



Transposed direct form II structure.

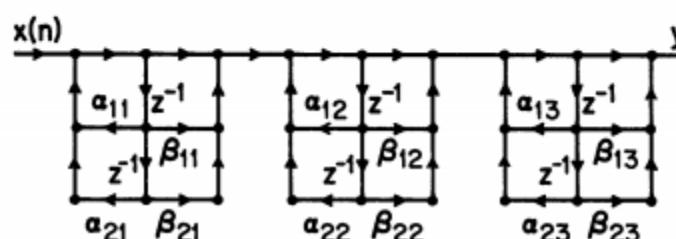
## Cascade Structure

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

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

Factorization of the z-transform for the cascade structure.

j.



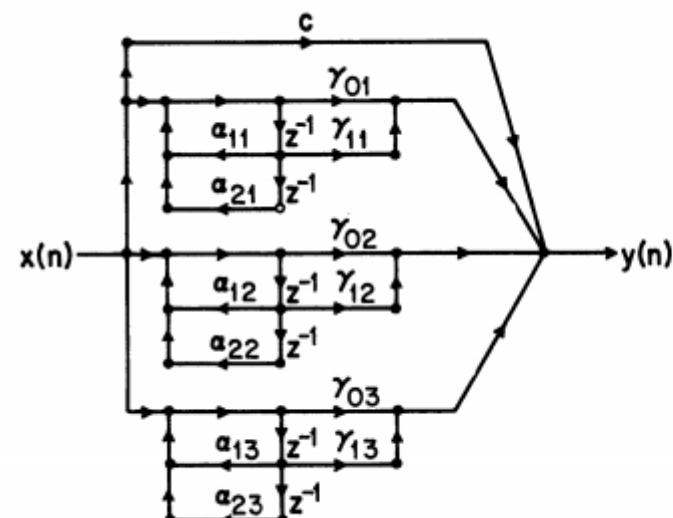
Cascade structure with a direct form II realization of each second-order subsystem.

# Class Review

k.

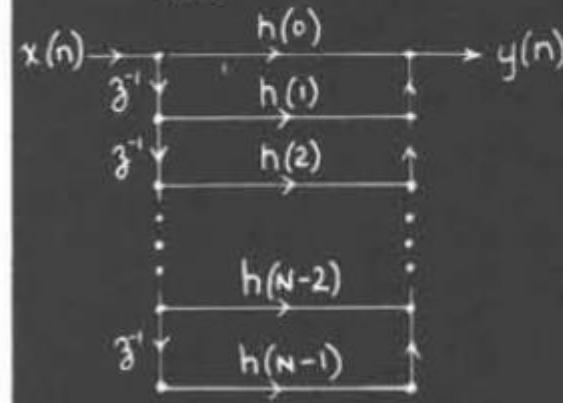
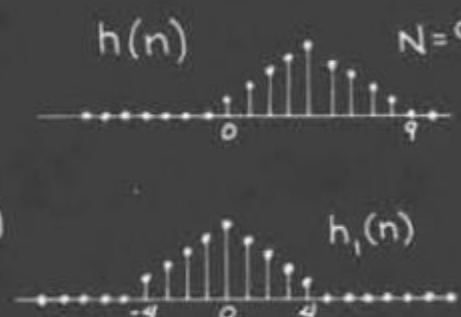
$$\begin{aligned}
 H(z) &= \frac{\sum_{k=0}^M b_k z^{-k}}{1 - \sum_{k=1}^N a_k z^{-k}} \\
 &= \sum_{k=1}^{N_1} \frac{A_k}{1 - c_k z^{-1}} + \sum_{k=1}^{N_2} \frac{B_k(1 - e_k z^{-1})}{(1 - d_k z^{-1})(1 - d_k^* z^{-1})} + \sum_{k=0}^{M-N} c_k z^{-k} \\
 &= \sum_{k=1}^{\frac{N+1}{2}} \frac{(\gamma_{0k} + \gamma_{1k} z^{-1})}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}} + \sum_{k=0}^{M-N} c_k z^{-k}
 \end{aligned}$$

Partial Fraction expansion for parallel structure.



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

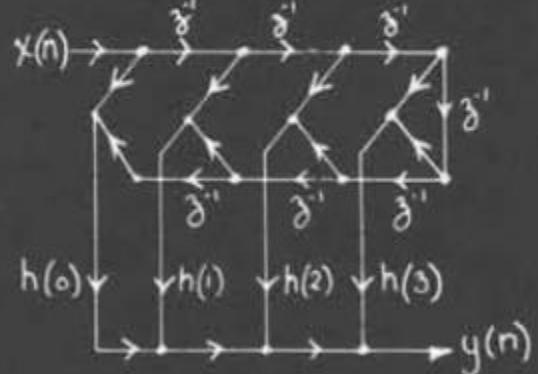
# Class Review

<p><b>FIR Systems</b></p> $H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$ $y(n) = \sum_{k=0}^{N-1} h(k) x(n-k)$ 	<p><b>Linear Phase FIR Systems</b></p> $h(n) = h(N-1-n)$  $h(n) = h_i\left(n - \frac{N-1}{2}\right)$ $H(e^{j\omega}) = e^{-j\omega\left(\frac{N-1}{2}\right)} H_i(e^{j\omega})$ $h_i(n) \text{ even} \Rightarrow H_i(e^{j\omega}) \text{ real}$	$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$ <p>assume N even</p> $= \sum_{n=0}^{\frac{N}{2}-1} h(n) z^{-n} + \sum_{n=\frac{N}{2}}^{N-1} h(n) z^{-n}$ $= \sum_{r=0}^{\frac{N}{2}-1} \underbrace{h(N-1-r)}_{h(r)} z^{-(N-1-r)}$ $= \sum_{n=0}^{\frac{N}{2}-1} h(n) z^{-n} + \sum_{n=0}^{\frac{N}{2}-1} h(n) z^{-(N-1-n)}$ $= \sum_{n=0}^{\frac{N}{2}-1} h(n) [z^{-n} + z^{-(N-1-n)}]$
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Class Review

$$H(z) = \sum_{n=0}^{\frac{N}{2}-1} h(n) [z^{-n} + z^{-(N-1-n)}]$$
  

$$N=8$$

$$H(z) = \sum_{n=0}^3 h(n) [z^{-n} + z^{-(7-n)}]$$


Frequency Sampling Structure  
 $h(n) \Leftrightarrow H(k)$  DFT  
 $H(z) \Leftrightarrow z\text{-transform}$

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

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

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

$$H(z) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) \underbrace{\sum_{n=0}^{N-1} (z^{-1} W_N^{-k})^n}_{\frac{1}{1 - z^{-N} W_N^{-k} z^{-1}}} \frac{1}{1 - W_N^{-k} z^{-1}}$$



# Class Review

Parameter Quantization

$$H(\bar{z}) = \frac{B(\bar{z}^{-1})}{A(\bar{z}^{-1})}$$

$$A(\bar{z}^{-1}) = 1 - \sum_{k=1}^n a_k \bar{z}^{-k}$$

$$= \prod_{k=1}^n (1 - z_k \bar{z}^{-1})$$

$$\hat{a}_k = a_k + \Delta_k$$

then

$$\hat{z}^i = z^i + \Delta z^i$$

$$\Delta z^i = \sum_{k=1}^n \left( \frac{\partial \bar{z}^i}{\partial a_k} \right) \Delta a_k$$

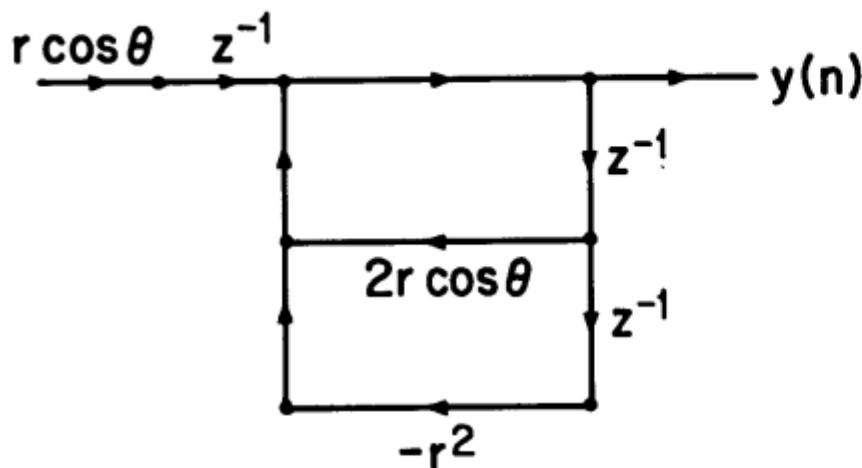
It can be shown that:

$$\frac{\partial \bar{z}^i}{\partial a_k} = \frac{\bar{z}^{(n-k)}}{\prod_{l=1, l \neq i}^n (\bar{z}_l - \bar{z}^i)}$$

$$\text{Let } |z^i - z^e| \leq \epsilon$$

$$\left| \frac{\partial \bar{z}^i}{\partial a_k} \right| \geq \frac{|z^i|^{n-k}}{\epsilon^{n-1}}$$

# Class Review

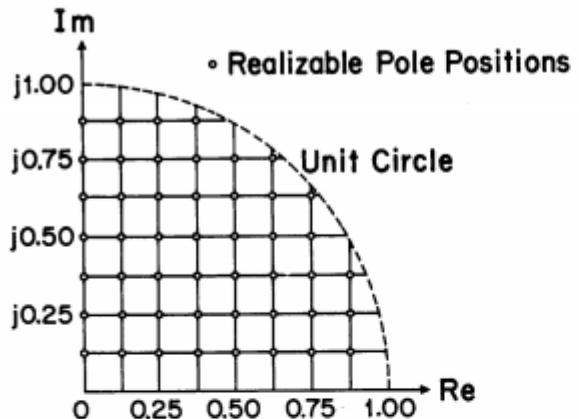


Direct-form implementation of a complex conjugate pole pair.

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

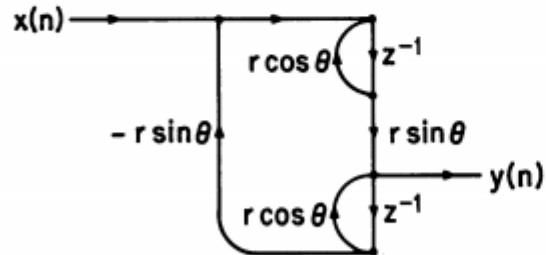
# Class Review

d.



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

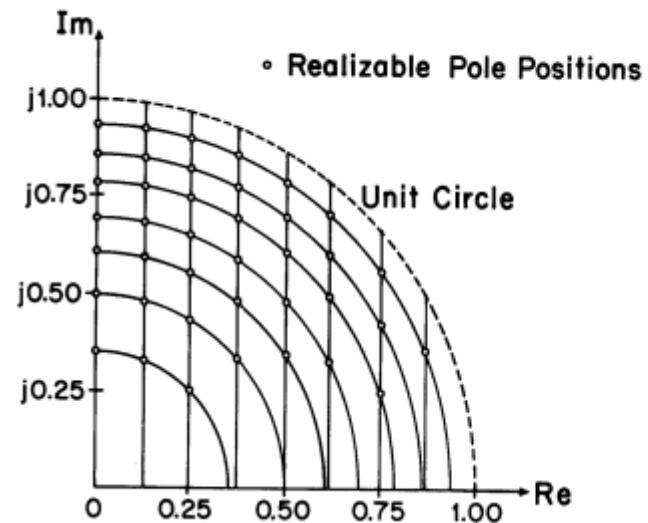
e.



Coupled form implementation of a complex conjugate pole pair.  
(Note that the transfer function has been corrected. The numerator factor is  $r \sin \theta$  not  $r \cos \theta$  as indicated in the lecture.)

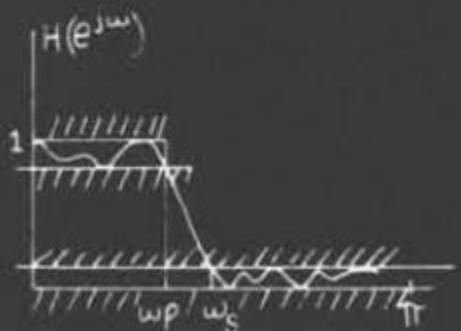
$$H(z) = \frac{r \sin \theta z^{-1}}{(1 - re^{j\theta} z^{-1})(1 - re^{-j\theta} z^{-1})}$$

# Class Review



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

# Class Review

<b>Digital Filter Design</b> $x(n) \rightarrow \boxed{\text{LSI}} \rightarrow y(n)$ $e^{j\omega_0 n} \rightarrow H(e^{j\omega_0}) e^{j\omega_0 n}$ $\cos \omega_0 n \rightarrow  H  \cos(\omega_0 n + \theta)$ 	<b>Design Techniques -</b> ① analytical ② continuous-time ③ algorithmic (computer-aided) <i>continuous → discrete</i> $H_a(s) \rightarrow H(z)$ $h_a(t) \rightarrow h(n)$ ① $j\Omega$ -axis (s-plane) $\rightarrow$ unit circle (z-plane) ② $H_a(s)$ Stable $\Rightarrow$ $H(z)$ Stable
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Class Review

$$\text{Differentials} \rightarrow \text{Differences} \quad \frac{d^k y_a(t)}{dt^k} \rightarrow \Delta^{(k)}[y(n)] \quad H(z) = H_a(s) \Big|_{s=\frac{z-1}{T}}$$

$H_a(s)$

$$\sum_{k=0}^n C_k \frac{d^k y_a(t)}{dt^k} = \sum_{k=0}^m d_k \frac{d^k x_a(t)}{dt^k}$$

$$\Delta^{(k)}[y(n)] = \Delta^{(1)}[\Delta^{(k-1)}y(n)]$$

$$s = \frac{\bar{z}^{-1}}{\tau} \quad \bar{z} = 1 + s\tau$$

$$\gamma_0(+) \rightarrow \gamma(n)$$

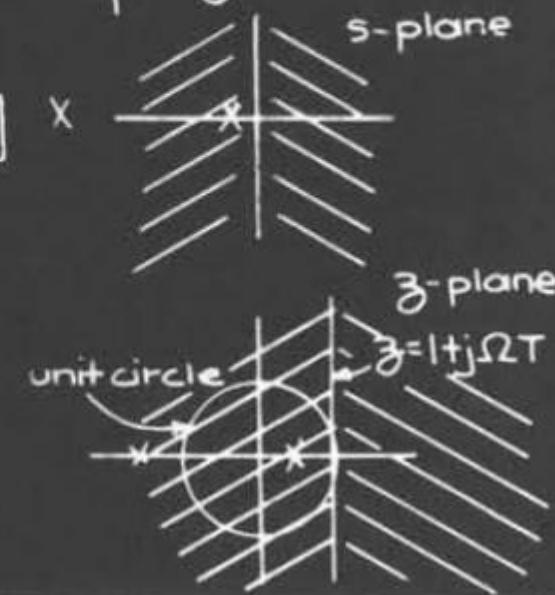
$$\frac{dy_a(t)}{dt} \Big|_{t=nT} \rightarrow \Delta^{(1)}[y(n)]$$

$$\sum_{k=0}^N c_k \Delta^{(k)}[y(n)] = \sum_{k=0}^M d_k \Delta^{(k)}[\chi(n)]$$

$$\mathcal{L}\left[\frac{dy_a(t)}{dt}\right] = sY_a(s)$$

$$\Delta^{(1)}[y(n)] = \frac{y(n+1) - y(n)}{T}$$

$$\sum \left[ \frac{Y(n+1) - Y(n)}{\tau} \right] = \frac{\bar{Y}-1}{\tau} Y(\bar{Y})$$



# Class Review

Impulse Invariance

$$h(n) = h_a(nT)$$

$$H(e^{j\omega}) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} H_a \left[ \frac{j\omega}{T} + \frac{j2\pi k}{T} \right]$$

$$H_a(s) = \sum_{k=1}^N \frac{A_k}{s - s_k}$$

$$h_a(t) = \sum_{k=1}^N A_k e^{s_k t} u(t)$$

$$h(n) = h_a(nT) = \sum_{k=1}^N A_k e^{s_k nT} u(n)$$

$$h(n) = \sum_{k=1}^N A_k (e^{s_k T})^n u(n)$$

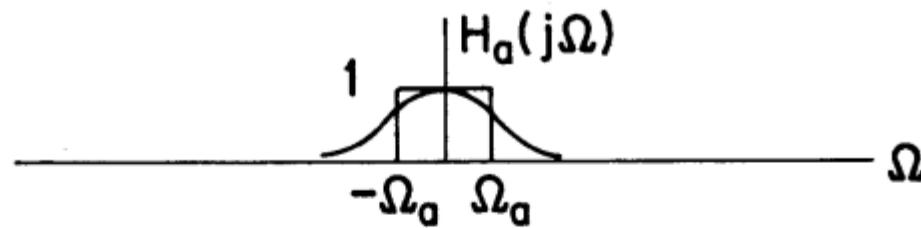
$$H(z) = \sum_{k=1}^N \frac{A_k}{1 - e^{s_k T} z^{-1}}$$

pole at  
 $s = s_k \implies z = e^{s_k T}$

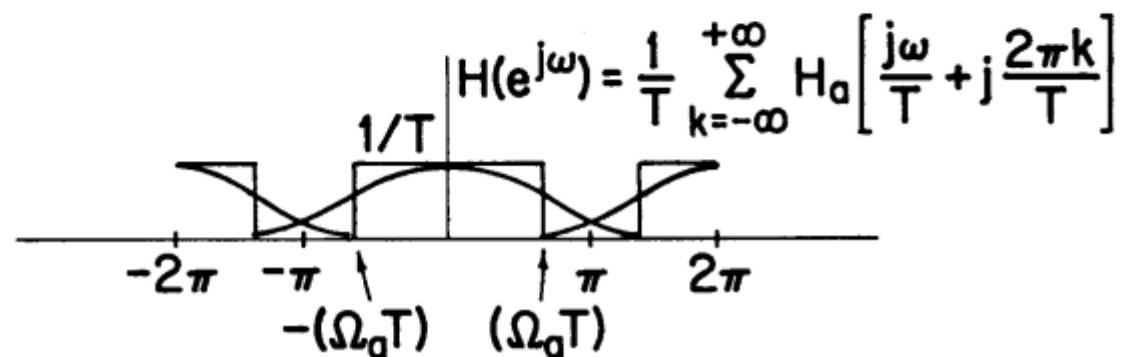
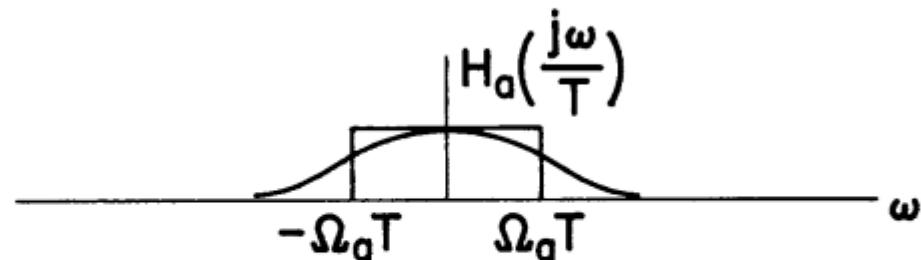
$$s_k = \sigma_k + j\omega_k \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Re}[s_k] < 0$$

$$|z_k| = |e^{\sigma_k T}| |e^{j\omega_k T}| \quad \left. \begin{array}{l} \\ \downarrow \\ |z_k| < 1 \end{array} \right\}$$

# Class Review



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



# Class Review

Differentials → Differences for  $z^k e^{j\omega} \rightarrow z^{-\frac{k}{2}} [e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}]$

$$s \rightarrow \frac{z+1}{T}$$

Impulse Invariance

$$\sum_{k=1}^N \frac{A_k}{s-s_k} \rightarrow \sum_{k=1}^N \frac{A_k}{1-e^{-\frac{1}{T}} z^{-\frac{1}{T}}}$$

Bilinear Transformation

$$H_a(s) \Rightarrow H(z)$$

$$s = \frac{2}{T} \left[ \frac{1-z^{-1}}{1+z^{-1}} \right]$$

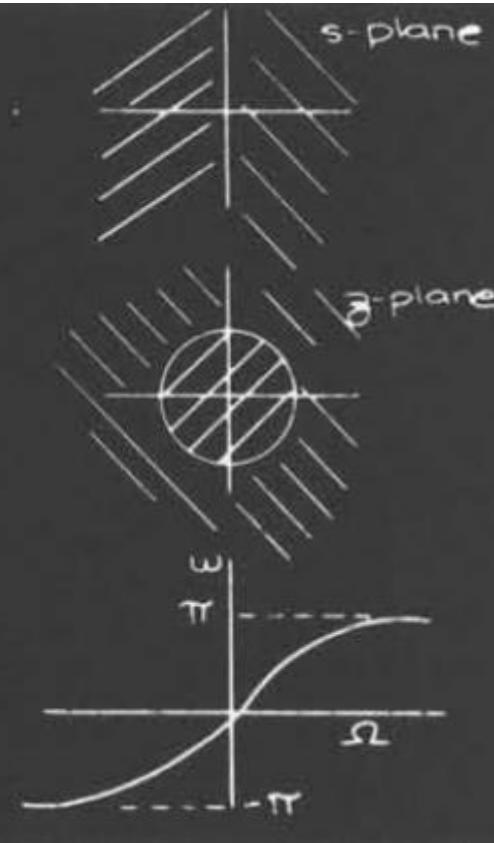
$$z = \frac{1 + \frac{j\omega}{2}}{1 - \frac{j\omega}{2}}$$

$$s = \frac{2}{T} \left[ \frac{1-e^{-j\omega}}{1+e^{-j\omega}} \right]$$

$$= \frac{2}{T} j \tan \frac{\omega}{2} = j\Omega$$

$$\Omega = \frac{2}{T} \tan \frac{\omega}{2}$$

$j\Omega$  axis  $\leftrightarrow$  unit circle



# Class Review

Algorithmic Design  
(IIR Filters)

① Minization of mean square error

$H_d(e^{j\omega})$  = Desired Frequency Response

$H_d(e^{j\omega_l}) \quad l=1, 2, \dots, M$

Error =

$$\sum_{l=1}^M [ |H(e^{j\omega_l})| - |H_d(e^{j\omega_l})| ]^2$$

$$H(z) = A \prod_{k=1}^K \frac{1 + a_k z^{-1} + b_k z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}}$$

choose parameters of  $H(z)$  to minimize E

Least Squares Inverse Design

Specify  $h_d(n)$

$$H(z) = \frac{b_0}{1 - \sum_{k=1}^N a_k z^{-k}} \leftrightarrow h(n)$$

$$h_d(n) \rightarrow \boxed{\frac{1}{H(z)}} \rightarrow g(n)$$

$$E = \sum_{n=0}^{\infty} [g(n) - \delta(n)]^2$$

$\Rightarrow$  Linear Equations

# Class Review

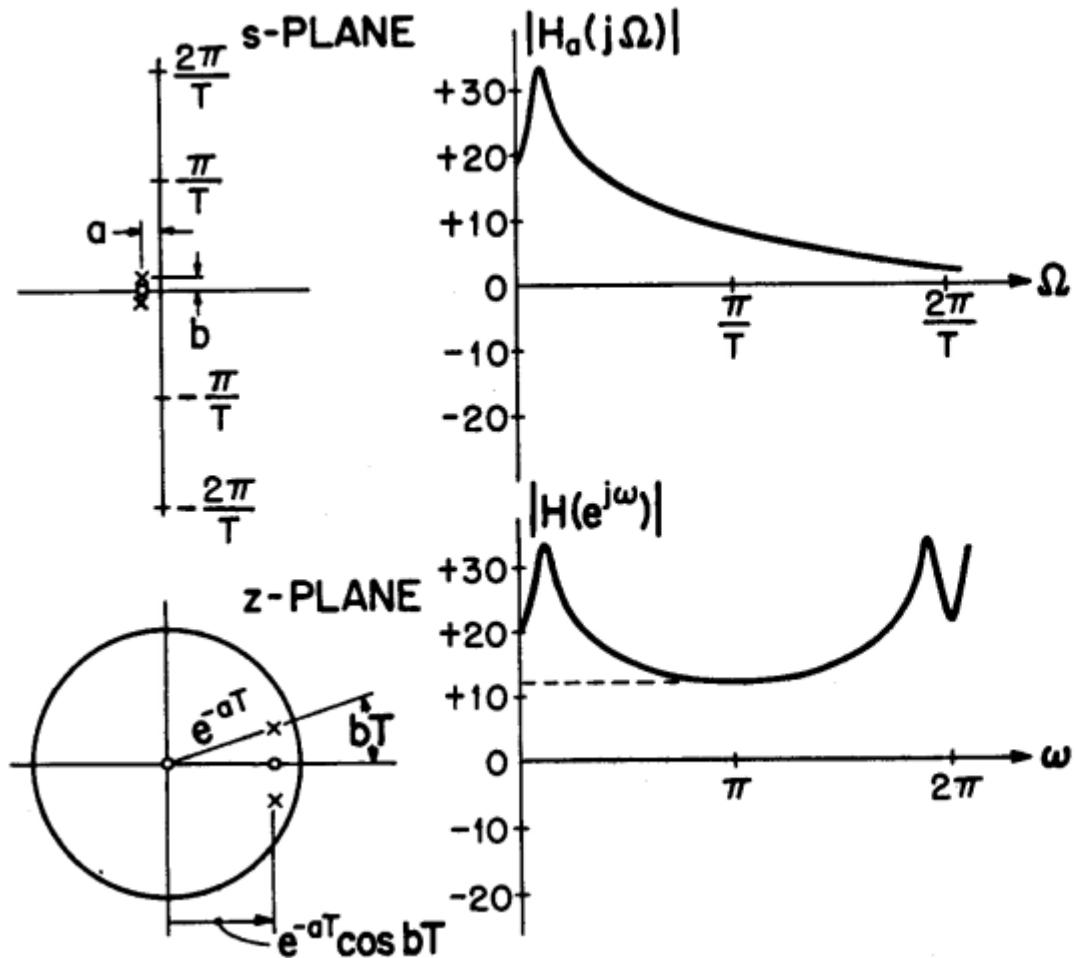
$$H_a(s) = \frac{(s+a)}{(s+a)^2 + b^2} = \frac{1/2}{s+a+jb} + \frac{1/2}{s+a-jb}$$

Example of impulse invariance.

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

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

# Class Review



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

# Class Review

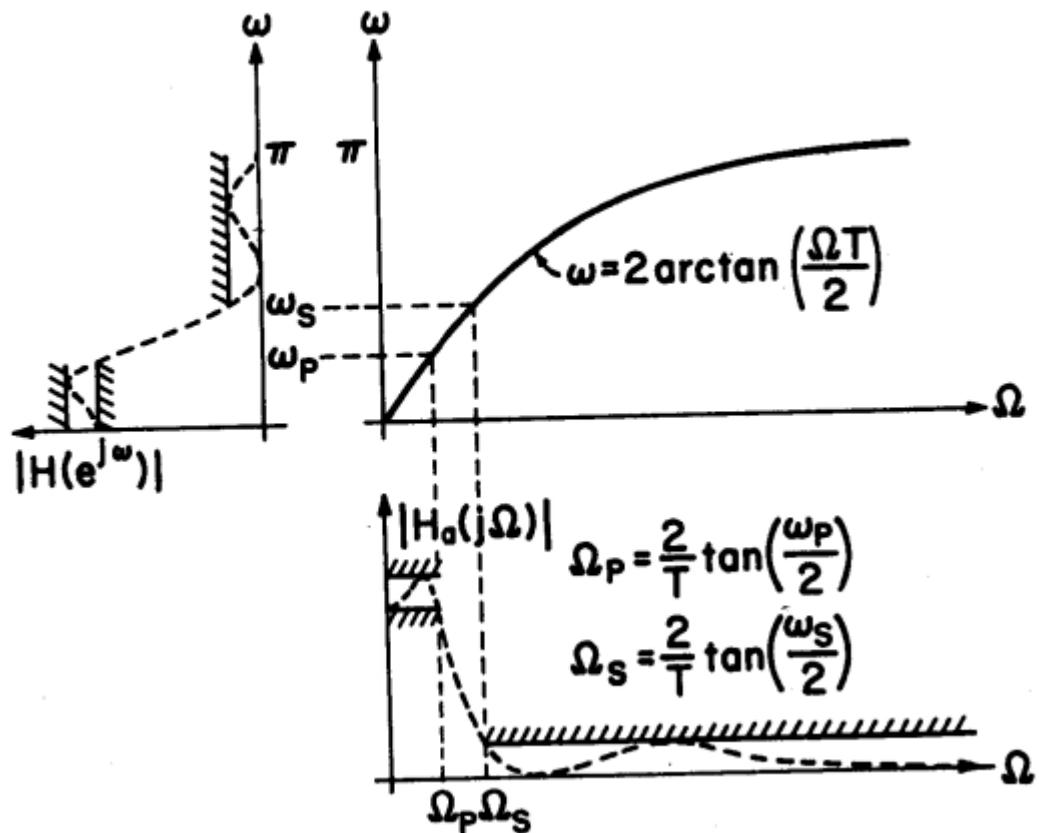
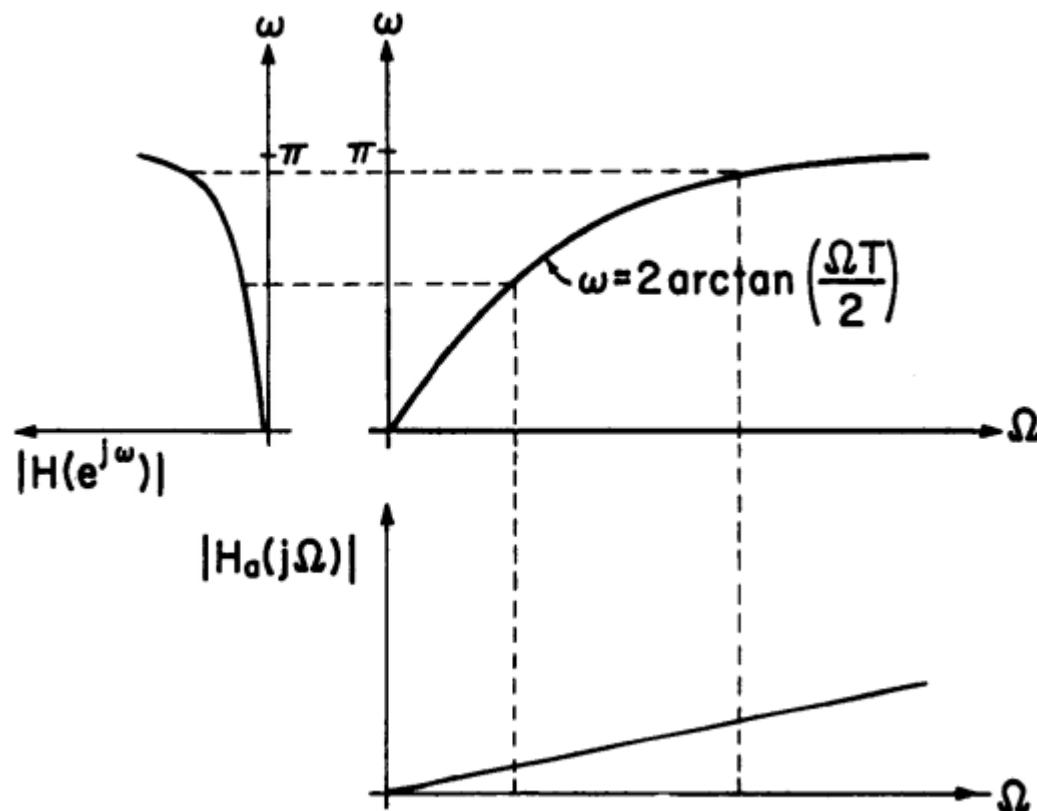


Illustration of effect  
of frequency warping  
inherent in the bi-  
linear transformation.

# Class Review



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

# Class Review

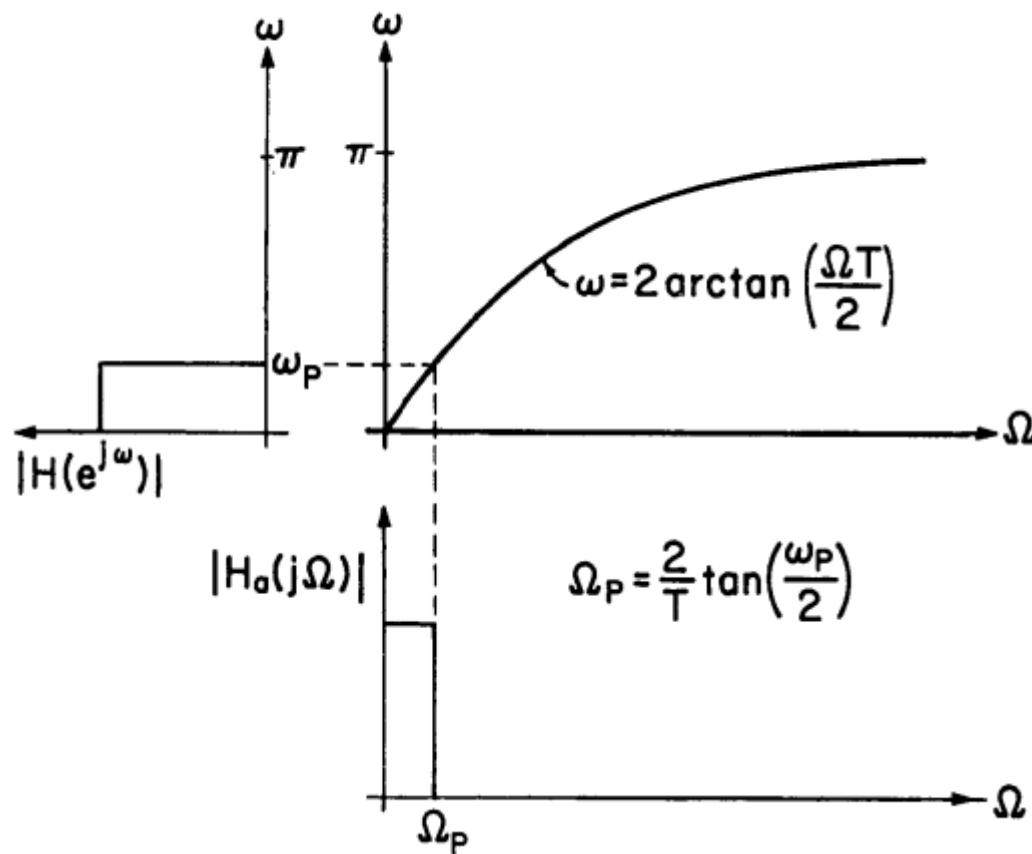
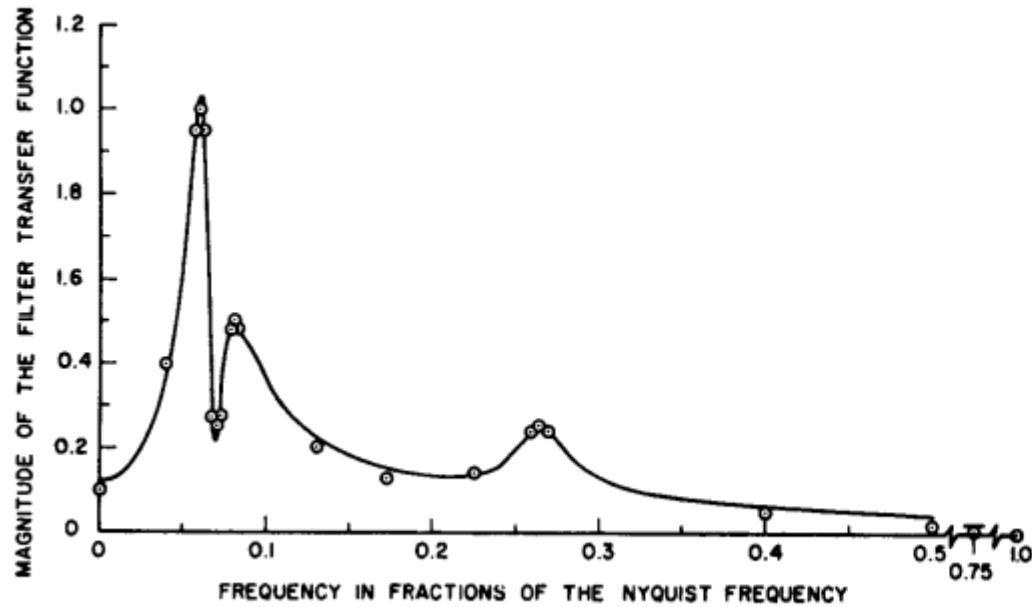


Illustration of effect  
of bilinear trans-  
formation on an equi-  
ripple frequency  
response characteristic.

# Class Review



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

# Class Review

## Design Examples

Analog Butterworth filter

$$|H_a(j\omega)|^2 = \frac{1}{1 + (\frac{\omega}{\omega_c})^{2N}}$$

$$|H_a(j\omega)|$$

N increases



$$H_a(S)H_a(-S) = \frac{1}{1 + (\frac{S}{j\omega_c})^{2N}}$$

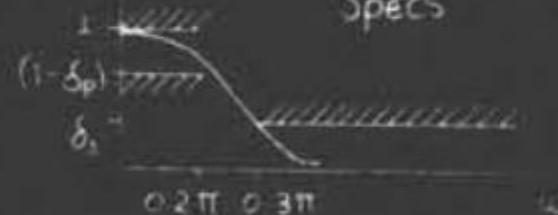
poles at

$$S_p = (-i)^{\frac{1}{2N}}(j\omega_c)$$

$$S_p = e^{j\left[\frac{\pi+2k\pi}{2N}\right]} e^{j\frac{\pi}{2}} \omega_c$$



Digital Filter Specs



$$(1 - \delta_p) \geq -1 \text{ db}$$

$$\delta_s \leq -15 \text{ db}$$

$$20 \log_{10} |H(e^{j\omega})| \geq -1$$

$$\text{or } |H(e^{j2\pi})| \geq 10^{-0.5}$$

also

$$20 \log_{10} |H(e^{-j\omega})| \leq -15$$

$$\text{or } |H(e^{-j\omega})| \leq 10^{-0.5}$$

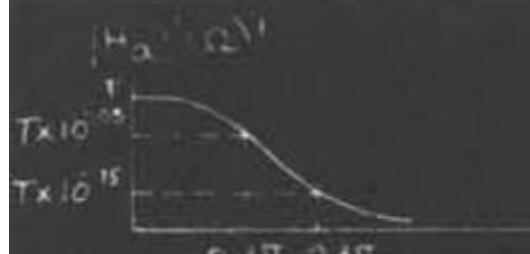
Impulse Invariant Design

$$h[n] = \sum_{k=-\infty}^{+\infty} h_a\left[\frac{n-k}{T} + j\frac{2\pi k}{T}\right]$$

$\Omega = \frac{\omega}{T}$

(reject Aliasing)

# Class Review



$$|H_a(j\omega)|^2 = \frac{T^2}{1 + \left(\frac{\omega}{\Omega_c}\right)^{2N}}$$

$$1 + \left(\frac{\frac{\pi}{T}}{\Omega_c}\right)^{2N} = 10^{-1}$$

$$1 + \left(\frac{\frac{3\pi}{T}}{\Omega_c}\right)^{2N} = 10^{-5}$$

$$N = 5.8858 \quad N = 6$$

$$\Omega_c T = 70474$$

choose  $N=6$

$$1 + \left(\frac{2\pi/T}{\Omega_c}\right)^{2 \times 6} = 10^{-1}$$

$$\Omega_c T = 7032$$

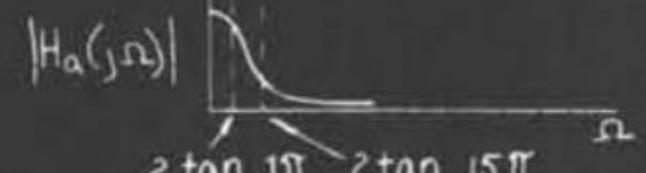
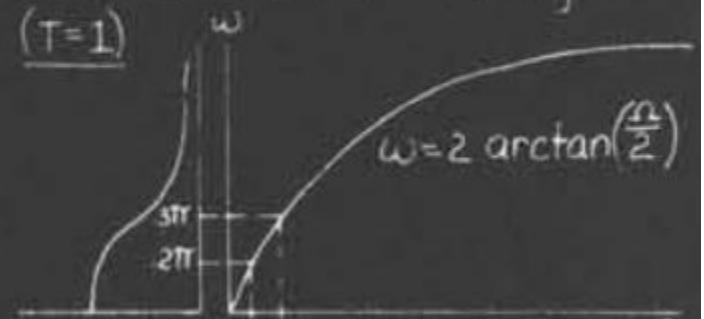
$$\underbrace{H_a(s) H_a(-s)}_{H_a(f)} = \frac{T^2}{1 + \left(\frac{2\pi}{7032}\right)^{2 \times 6}}$$

$$\frac{T=1}{h_a(f)} \longleftrightarrow H_a(s)$$

$$h_a(\frac{k}{T}) \longleftrightarrow T H_a(sT)$$

$$h(n) = h_a(nT) = h_a(n)$$

Bilinear Transform Design



$$20 \log_{10} |H_a(j2\tan.1\pi)| \geq -1$$

$$20 \log_{10} |H_a(j2\tan.15\pi)| \leq -15$$

# Class Review

$$| + \left[ \frac{j2\tan(0.1\pi)}{j\Omega_c} \right]^{2N} = 10^1 \quad ①$$

$$| + \left[ \frac{j2\tan(0.15\pi)}{j\Omega_c} \right]^{2N} = 10^{1.5} \quad ②$$

$$\Rightarrow N = 5.30466$$

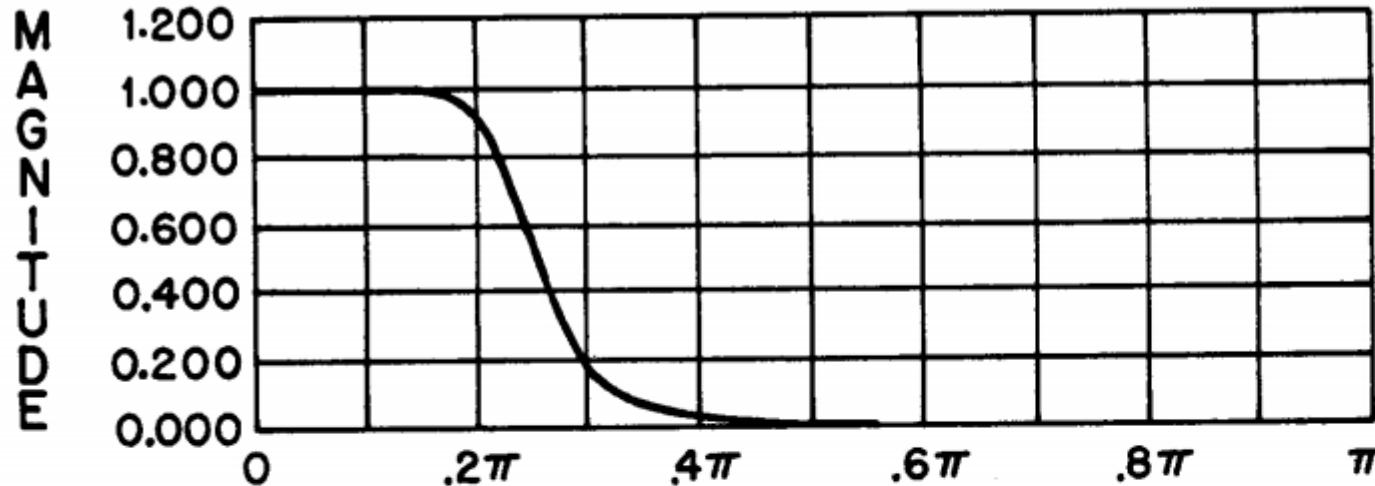
choose  $N=6$   $\underbrace{H_a(s) H_a(s)}$

Meet stopband  
exceed passband

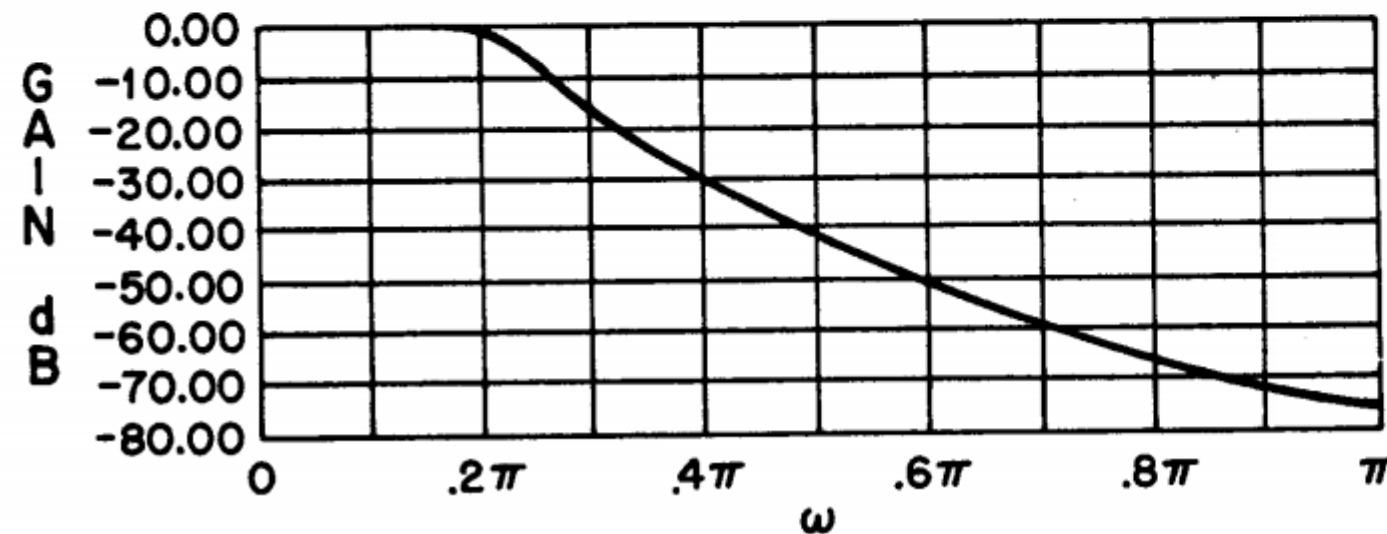
$$| + \left[ \frac{j2\tan(0.15\pi)}{j\Omega_c} \right]^{2 \times 6} = 10^{1.5}$$

$$\Rightarrow \Omega_c = 7662.2$$

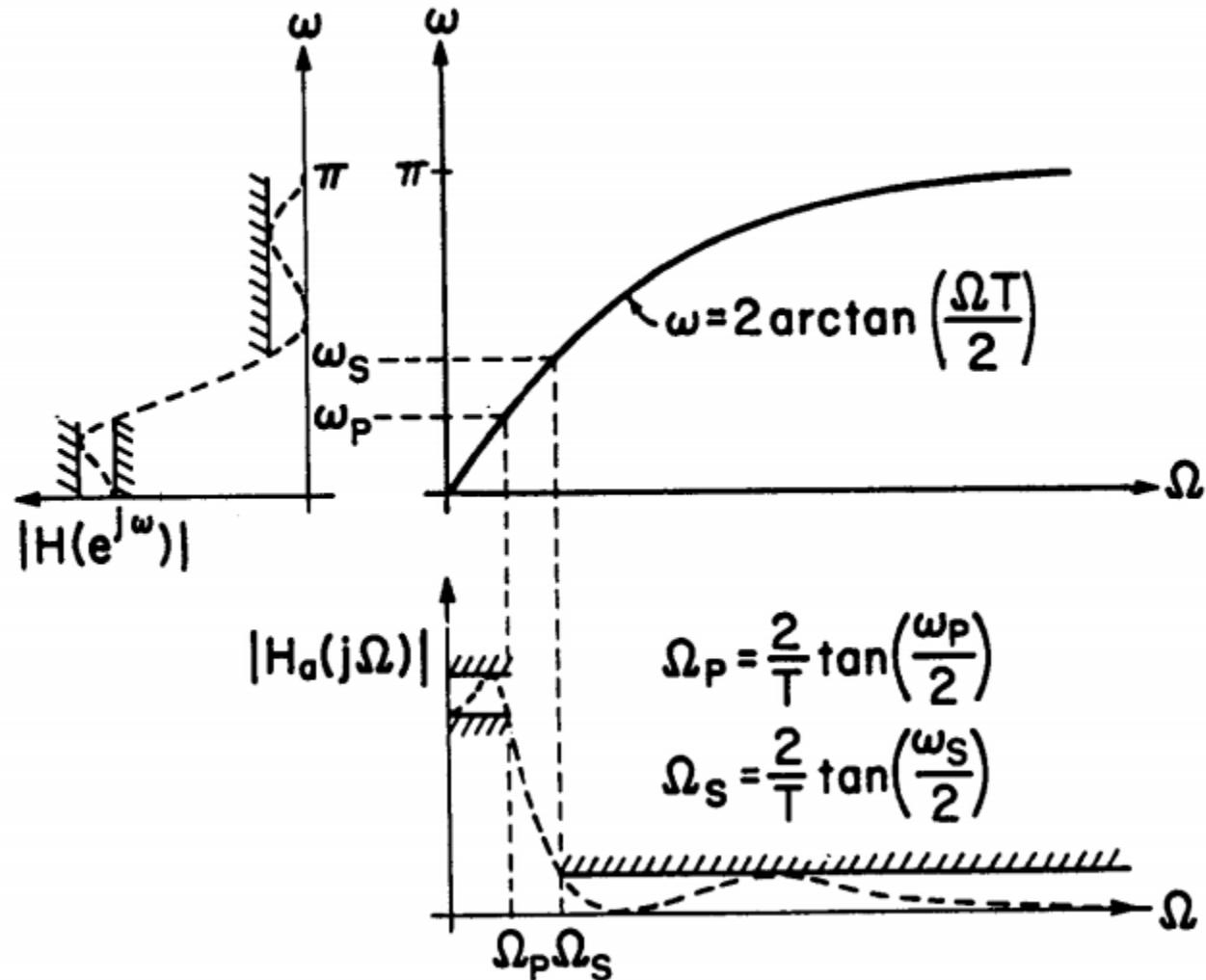
# Class Review



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

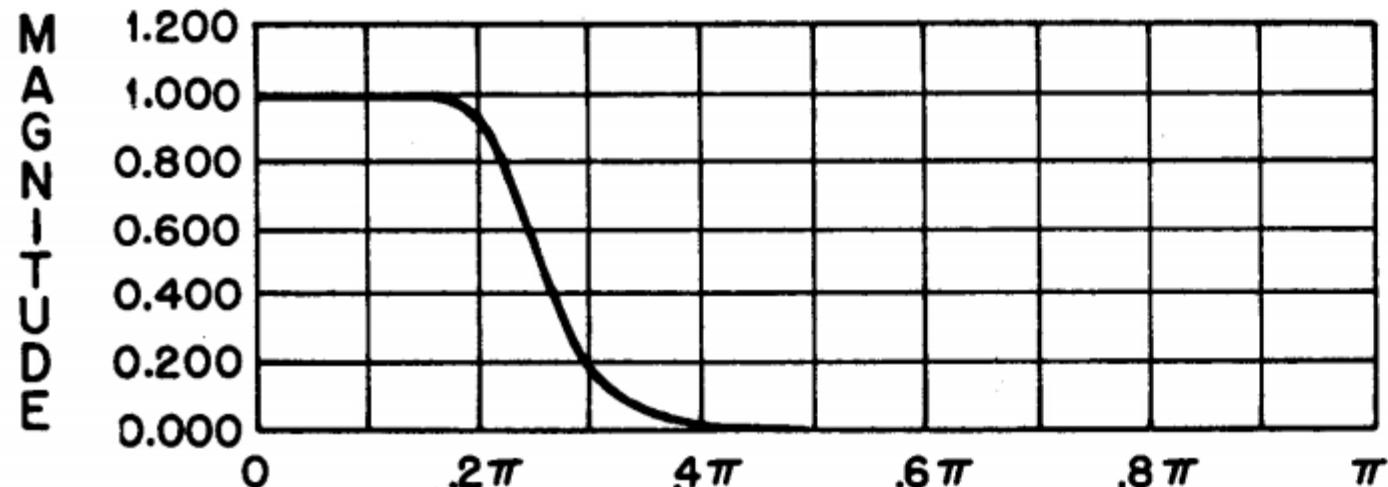


# Class Review

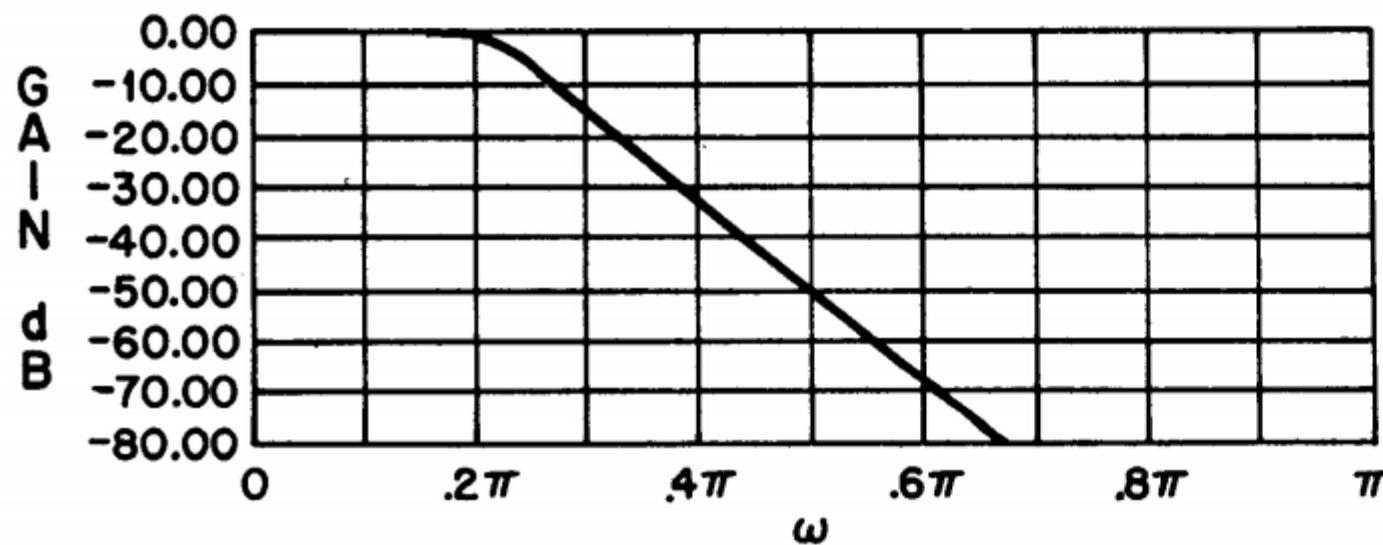


The bilinear trans-  
formation.

# Class Review



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



# Class Review

## DESIGN OF FIR DIGITAL FILTERS

Basic design methods  
for 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

a.

### DESIGN OF FIR FILTERS USING WINDOWS

Design of FIR filters  
using the window  
method.

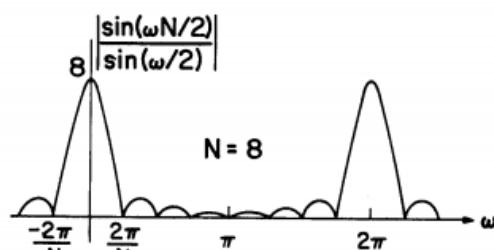
DESIRED UNIT SAMPLE RESPONSE:  $h_d(n)$

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

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

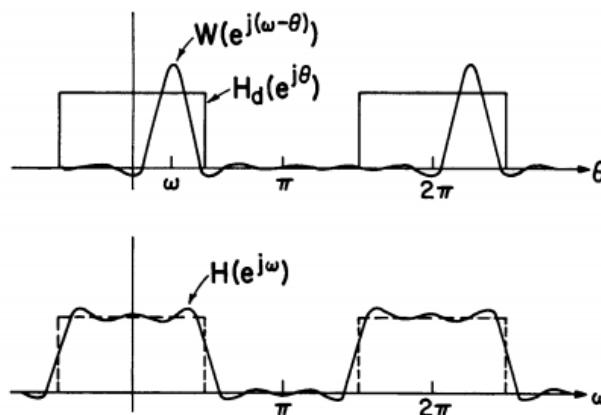
$$H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\theta}) W[e^{j(\omega-\theta)}] d\theta$$

b.

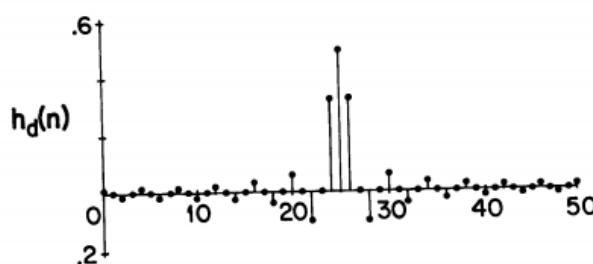


Magnitude of the  
Fourier transform for  
an eight point  
rectangular window.

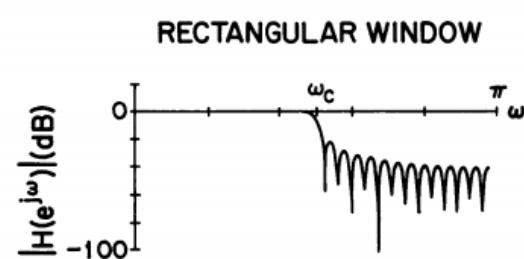
# Class Review



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

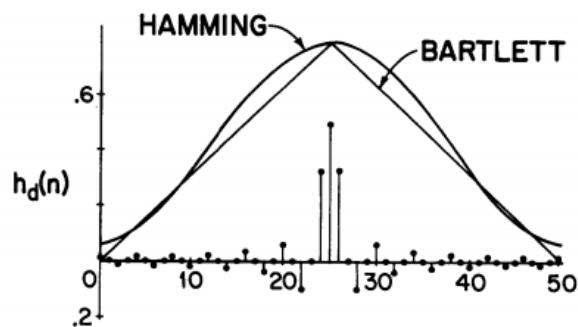


Unit-sample response of an ideal low-pass filter truncated by a 51-point rectangular window.



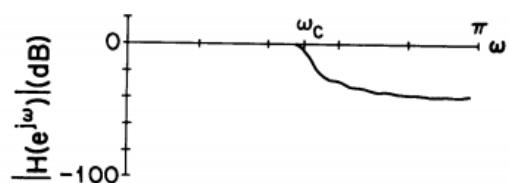
Frequency response corresponding to the unit-sample response in viewgraph e.

# Class Review



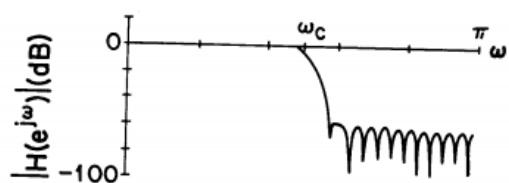
The Hamming and Bartlett windows.

BARTLETT WINDOW



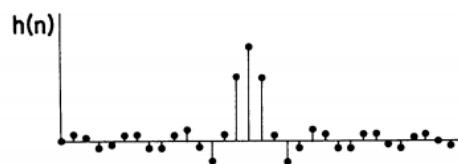
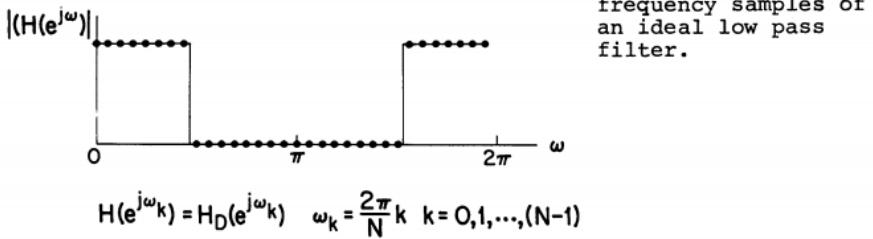
Frequency response of an FIR lowpass filter obtained by multiplying the unit-sample response of an ideal low pass filter by a Bartlett window.

HAMMING WINDOW

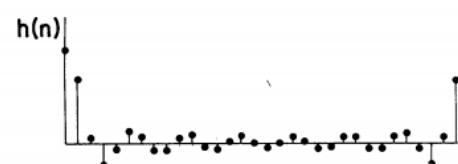


Frequency response of an FIR lowpass filter obtained by multiplying the unit sample response of an ideal lowpass filter by a Hamming window. (Note that the stopband attenuation is approximately 65 db not 30 db as stated in the lecture.)

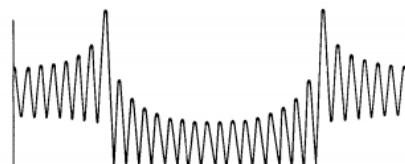
# Class Review



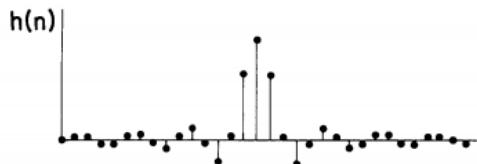
A unit-sample response whose DFT is equal to the frequency samples in viewgraph j.. The bottom trace is the magnitude of the Fourier transform of this unit-sample response.



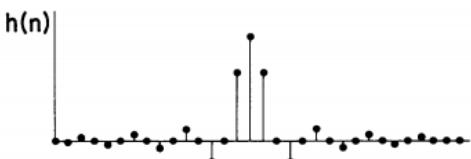
Another unit-sample response the magnitude of whose DFT is equal to the frequency samples in viewgraph j.. The bottom trace is the magnitude of the Fourier transform of this unit-sample response.



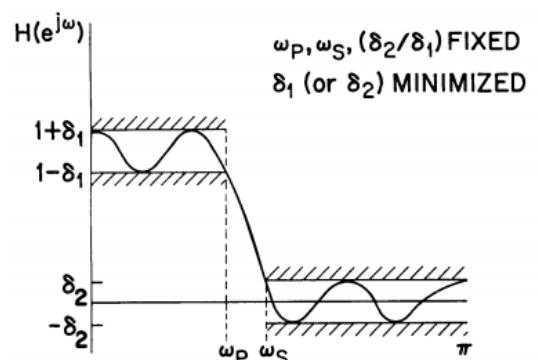
# Class Review



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

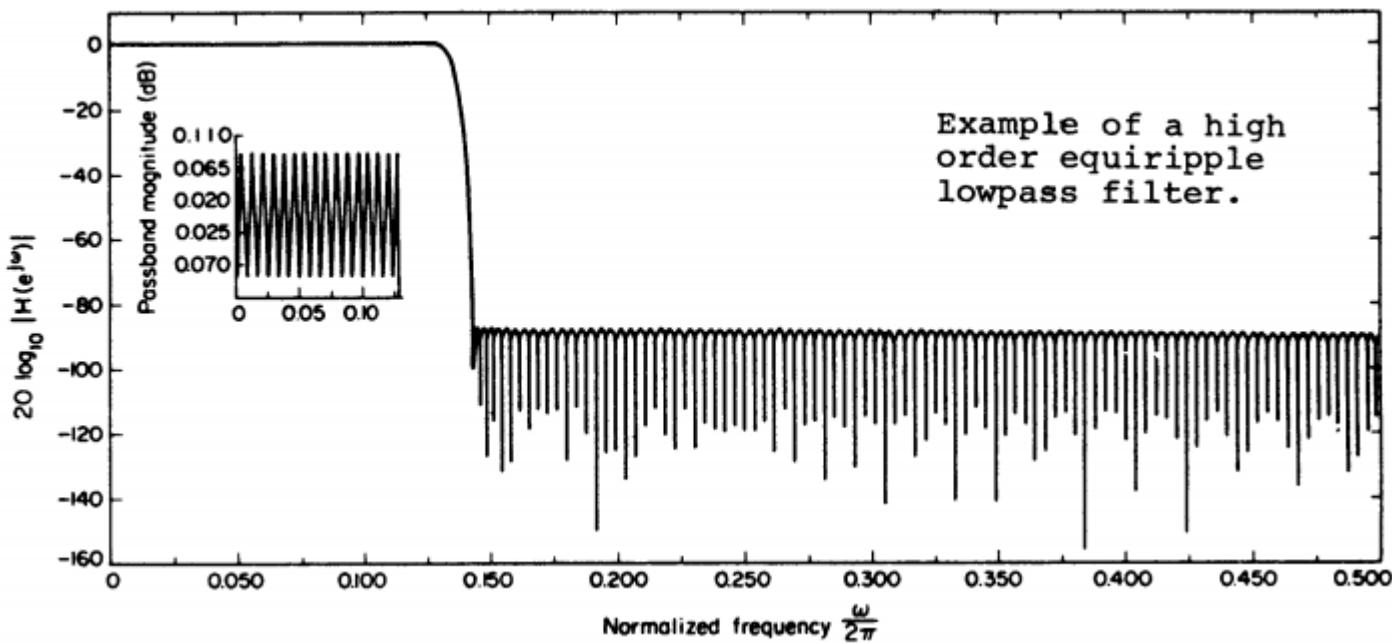


Similar to viewgraph m. with a different value of the frequenc sample in the transition band.



Equiripple approxi-  
mation of a lowpass  
filter.

# Class Review



# Class Review

Computation of the DFT

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

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

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

Direct Computation

$$X(k) W_N^{kn} \Rightarrow \begin{array}{l} 1 \text{ complex multiply} \\ (4 \text{ mults, } 2 \text{ adds}) \end{array}$$

$$X(k) \quad k=0, 1, \dots, N-1$$

$N^2$  complex multiplies

$N(N-1)$  complex adds

$\approx N^2$  MADS

Fast Fourier Transform (FFT)

$$N = P_1 P_2 \dots P_v$$

Complex MADS  $\propto$

$$N[P_1 + P_2 + \dots + P_v]$$

$$N = 2^v \Rightarrow N \log_2 N$$

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

$$= \underbrace{\sum_{\substack{n \text{ even} \\ n=2r}} x(n) W_n^{nk}}_{\sum_{\substack{n \text{ odd} \\ n=2r+1}}} + \underbrace{\sum_{\substack{n \text{ odd} \\ n=2r+1}} x(n) W_n^{nk}}$$

$$r = 0, 1, \dots, \frac{N}{2} - 1$$

# Class Review

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} \chi(2r) W_N^{2rk}$$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} \chi(2r) W_{\frac{N}{2}}^{rk}$$

$$+ \sum_{r=0}^{\frac{N}{2}-1} \chi(2r+1) W_N^{(2r+1)k}$$

$$+ W_N^k \sum_{r=0}^{\frac{N}{2}-1} \chi(2r+1) W_{\frac{N}{2}}^{rk}$$

$$W_N^{(2r+1)k} = W_N^k W_N^{2rk}$$

$$W_N^2 = e^{-j\frac{2\pi}{N} \cdot 2} = e^{-j\frac{2\pi}{N/(N/2)}}$$

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

# Class Review

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

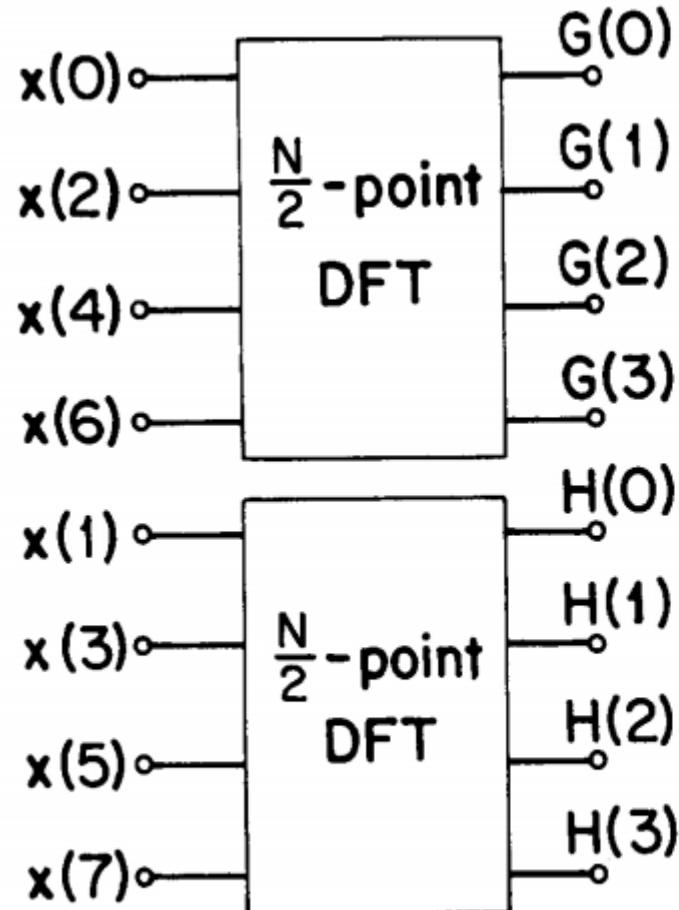
G(k)

$$\underline{X}(k) = G(k) + W_N^k H(k)$$

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

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

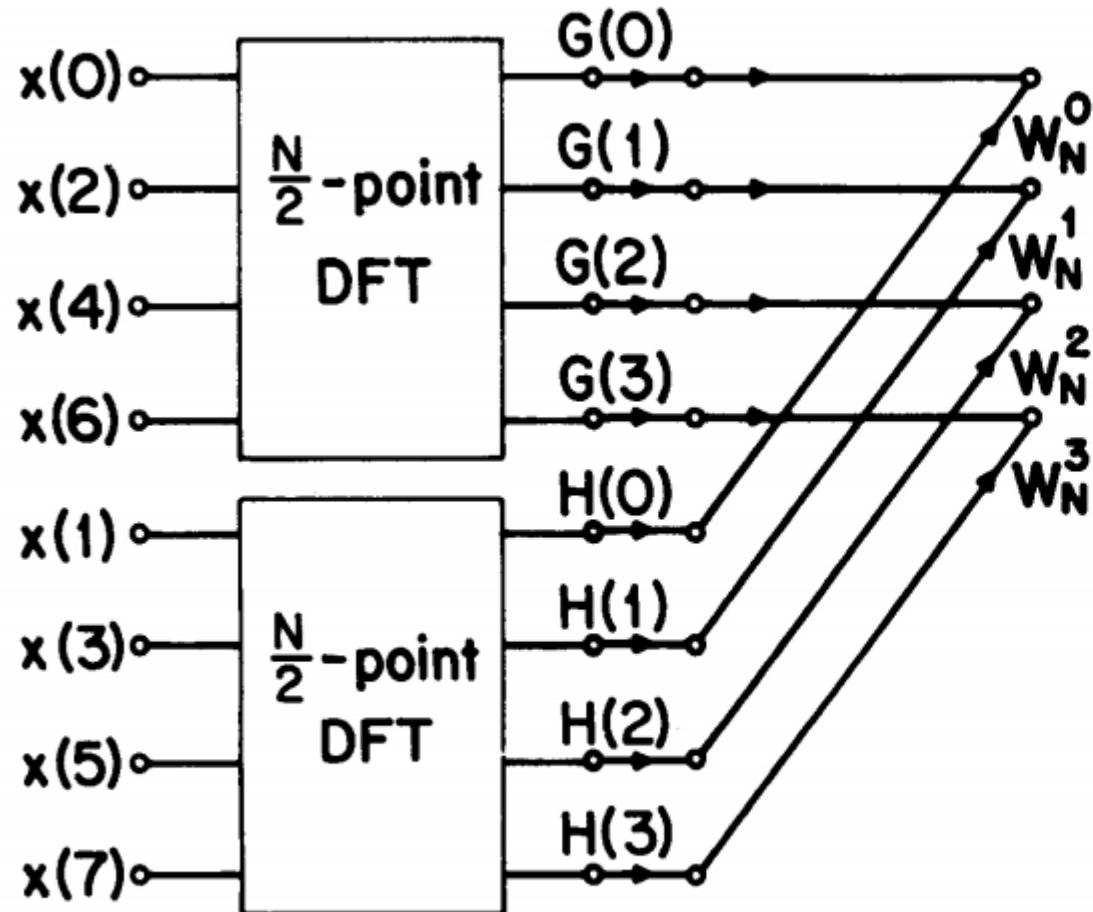
# Class Review



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

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

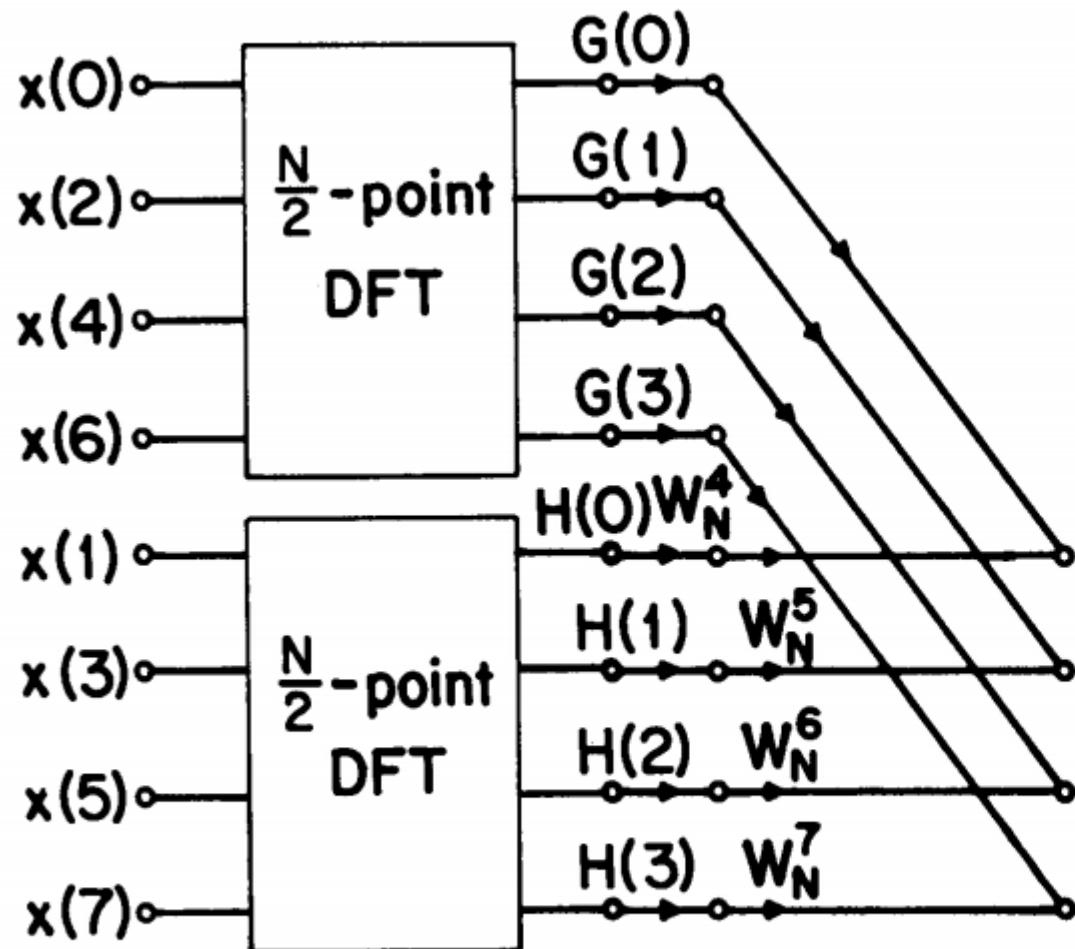
# Class Review



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

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

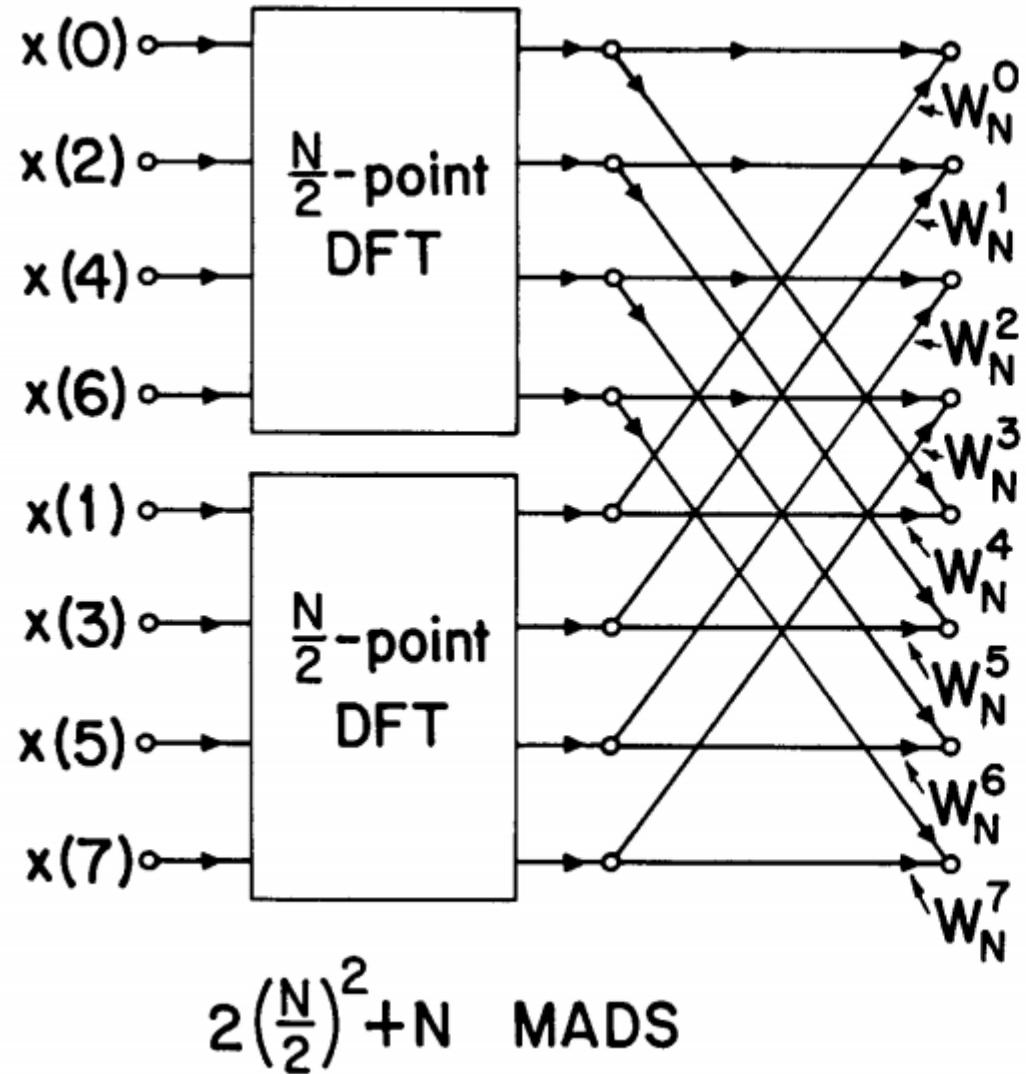
# Class Review



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

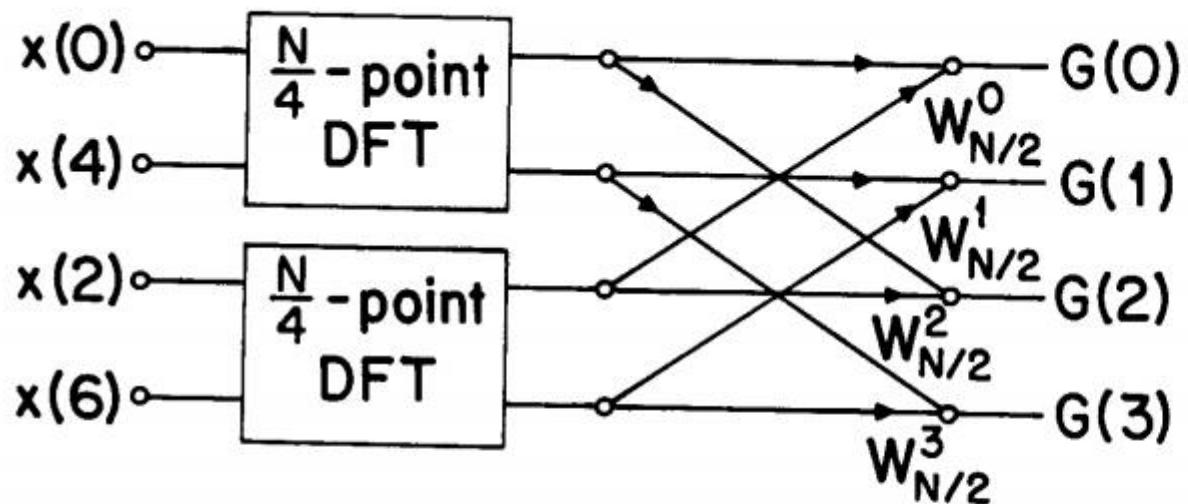
$$X(k) = G(k) + W_N^k H(k) \quad G(k+4) = G(k) \quad H(k+4) = H(k)$$

# Class Review



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

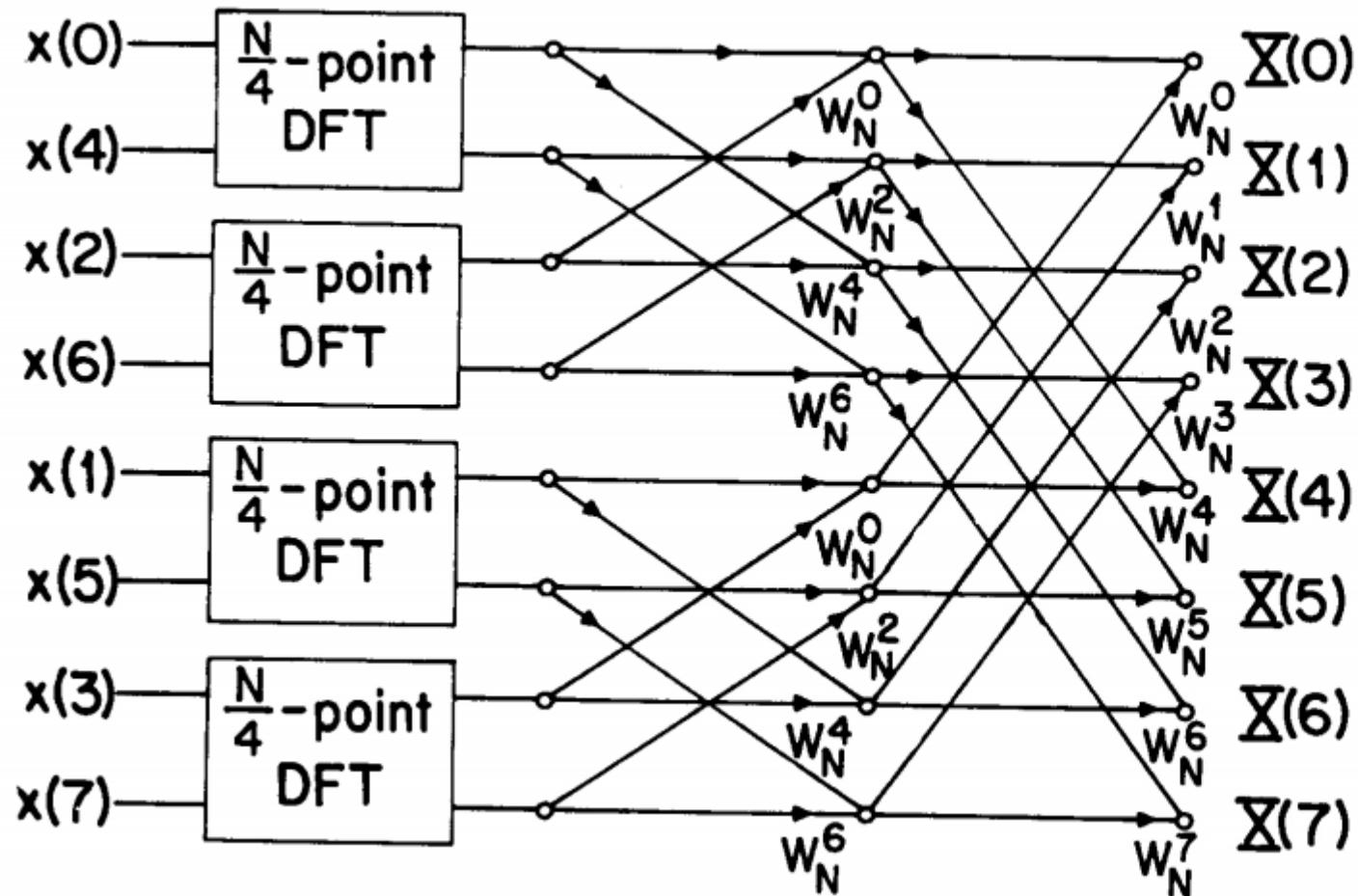
# Class Review



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

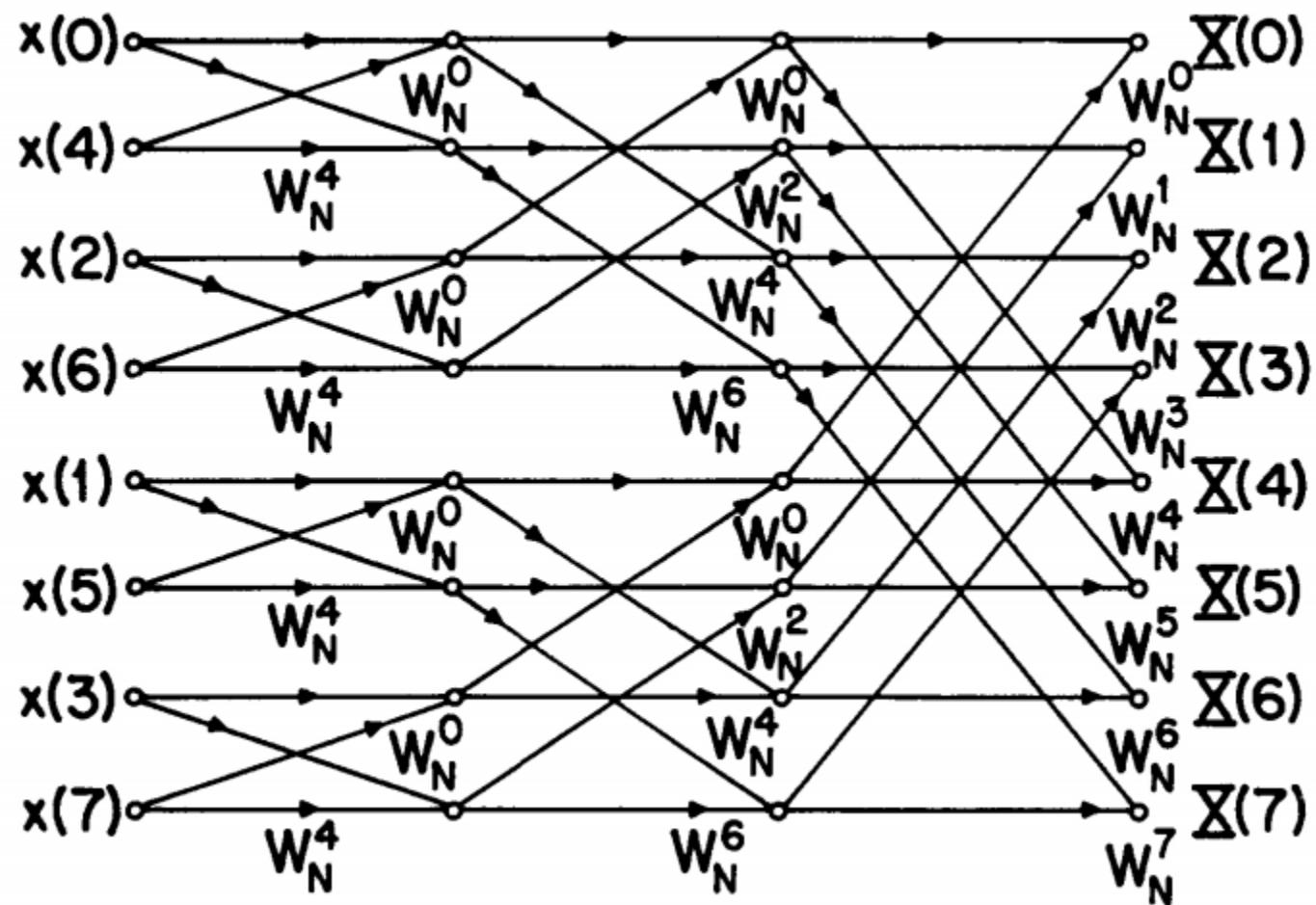
$$W_{\frac{N}{2}} = W_N^2$$

# Class Review



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

# Class Review



Flow-graph for computa-  
tion of  $X(k)$

# Class Review

$$N = 2^v$$

1 N POINT DFT

$$\downarrow N^2$$

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 +

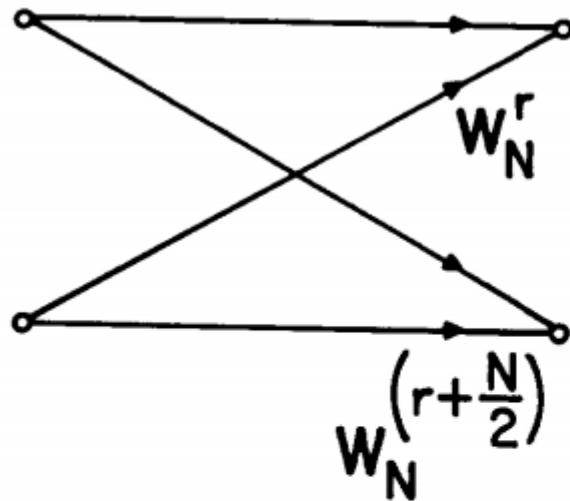
$$8\left(\frac{N}{8}\right)^2 + N + N + N$$

$$\left(\frac{N}{8}\right)^2 \rightarrow 2\left(\frac{N}{16}\right)^2 + \frac{N}{8}$$

$\underbrace{N+N+\cdots+N}_{v \text{ TIMES}}$

Savings in computation resulting from successive decimation in time

# Class Review

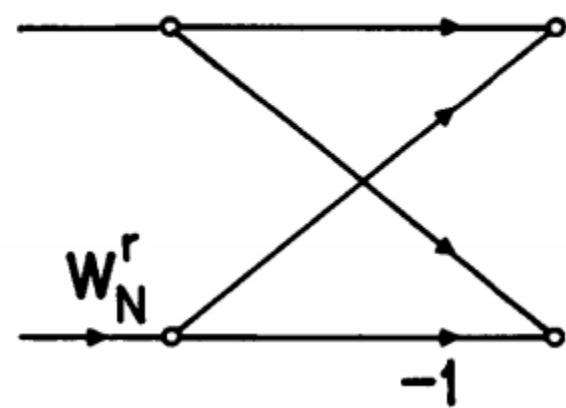


**Basic butterfly computation in flow-graph j.**

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

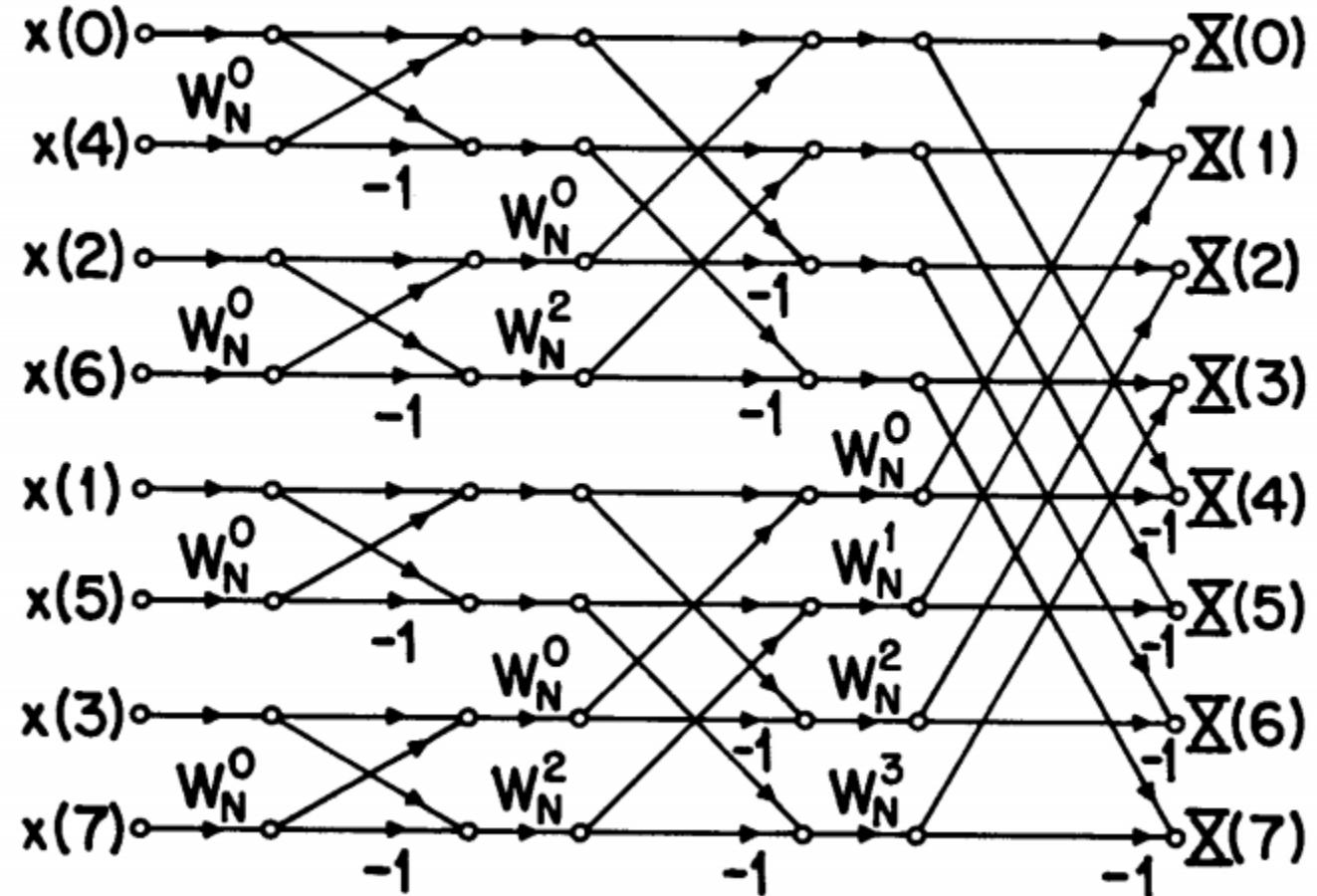
$$W_N^{\frac{N}{2}} = e^{-j \frac{2\pi}{N} \frac{N}{2}} = e^{-j\pi} = -1$$

# Class Review



Rearrangement of the butterfly computation in 1.

# Class Review



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

# Class Review

Decimation in Frequency

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

$$= \sum_{n=0}^{\frac{N}{2}-1} \chi(n) W_N^{nk} + \underbrace{\sum_{n=\frac{N}{2}}^{N-1} \chi(n) W_N^{nk}}_{W_N^{\frac{N}{2}k} \sum_{n=0}^{\frac{N}{2}-1} \chi(n+\frac{N}{2}) W_N^{nk}}$$

$$X(k) = k \cdot \text{odd-} X(2r+1)$$

$$\sum_{n=0}^{\frac{N}{2}-1} [\chi(n) + (-1)^k \chi(n+\frac{N}{2})] W_N^{nk} \quad r=0, 1, \dots, (\frac{N}{2}-1)$$

$$\sum_{n=0}^{\frac{N}{2}-1} [\chi(n) - \chi(n+\frac{N}{2})] W_N^n W_N^{2rn}$$

$$k \text{ even-} X(2r)$$

$$\sum_{n=0}^{\frac{N}{2}-1} [\chi(n) + \chi(n+\frac{N}{2})] W_N^{2rn} \quad W_N^{2rn} = W_{\frac{N}{2}}^{rn}$$

$$r=0, 1, 2, \dots, (\frac{N}{2}-1)$$

# Class Review

$k$  even

$$X(2r) = \sum_{n=0}^{\frac{N}{2}-1} g(n) W_n^{rn}$$

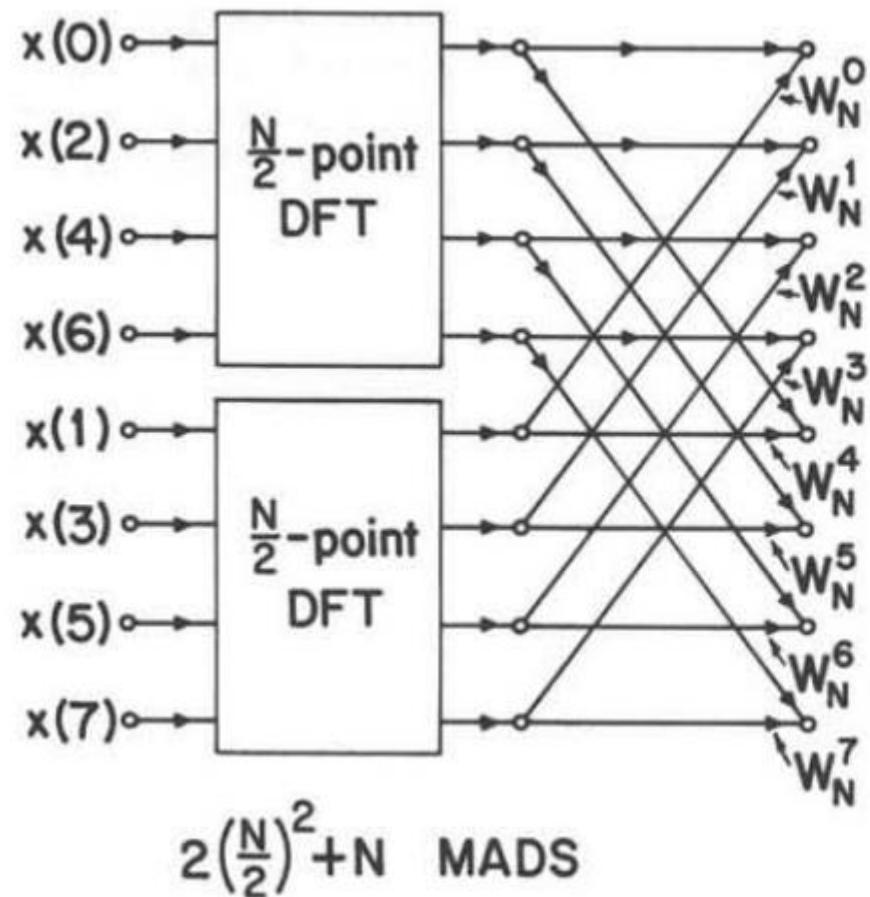
$$g(n) = \chi(n) + \chi(n + \frac{N}{2})$$

$k$  odd

$$X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} [h(n) W_n^n] W_{n/2}^{rn}$$

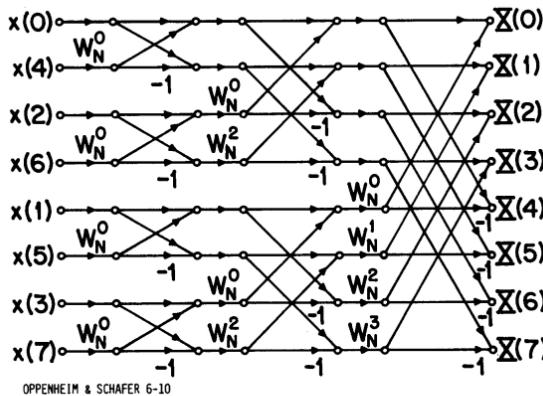
$$h(n) = \chi(n) - \chi(n + \frac{N}{2})$$

# Class Review



Decomposition of an  
 $N$ -point DFT into  
 $2 \frac{N}{2}$ -point DFT's.

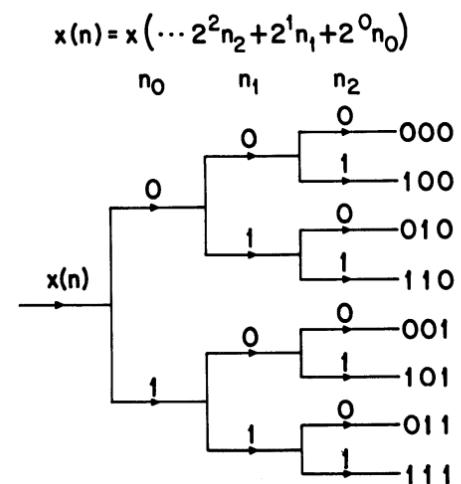
# Class Review



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

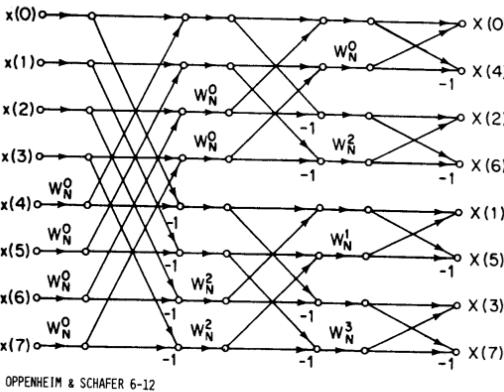
	Storage Register	Data Index	
0	000	$X(0)$	000
1	001	$X(4)$	100
2	010	$X(2)$	010
3	011	$X(6)$	110
4	100	$X(1)$	001
5	101	$X(5)$	101
6	110	$X(3)$	011
7	111	$X(7)$	111

Relation between data index and data storage register.

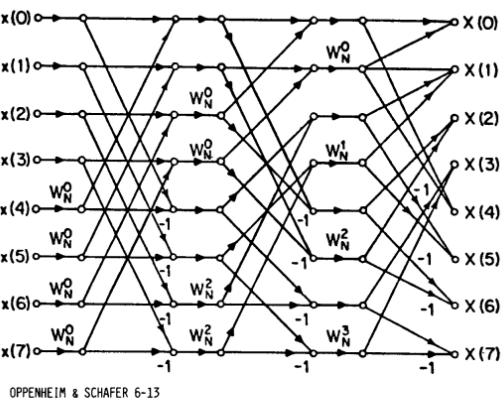


Sorting of data implied by the development of the decimation-in-time algorithm.

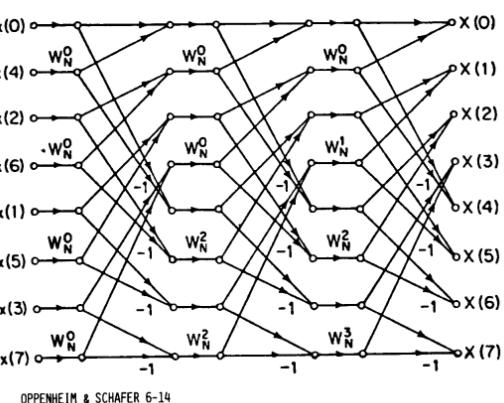
# Class Review



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

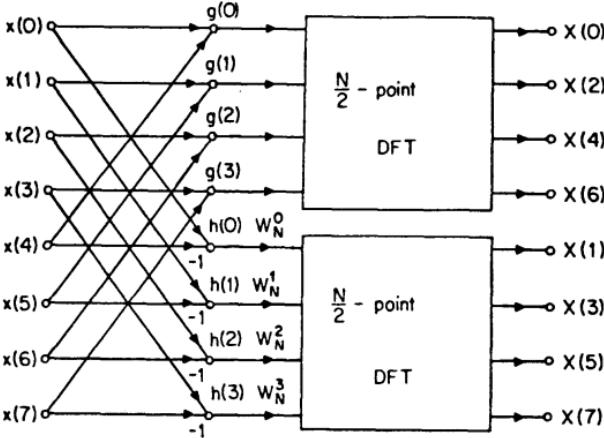


Rearrangement of flow-graph d. with both input and output in normal order.

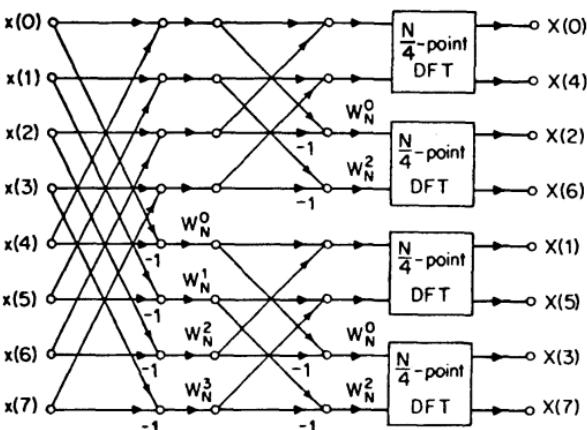


Rearrangement of flow-graph d having the same geometry for each stage.

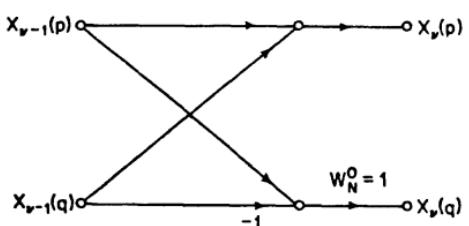
# Class Review



Computation of even  
and odd-numbered  
DFT values.

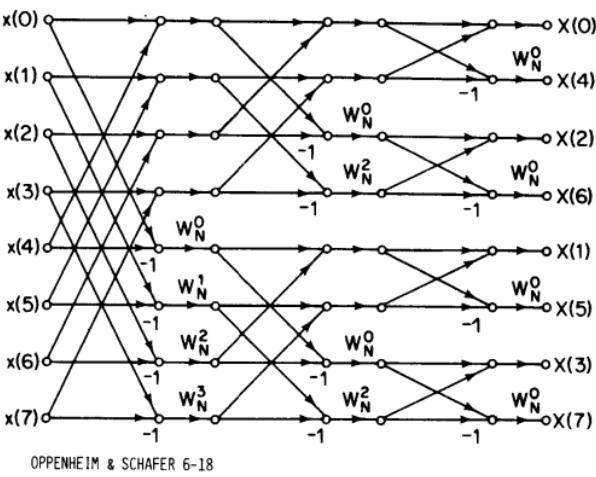


Decomposition of the  
 $N/2$ -point DFT's of  
flow-graph  $j$ .



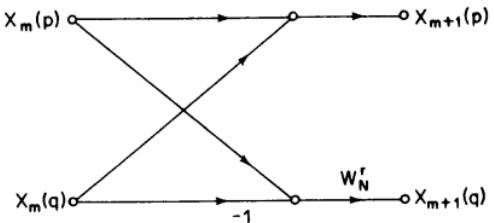
Flow-graph for two-  
point DFT.

# Class Review

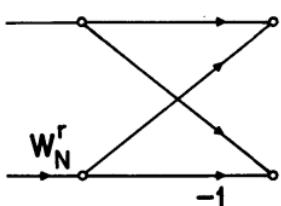


OPPENHEIM & SCHAFER 6-18

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

Computational Considerations

① Inverse DFT

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

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

DFT

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

② Bit Reversal



# 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  $q_1$



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

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

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

$$X(k) =$$

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

$$W_N^{P_1 r k} = e^{-j \frac{2\pi}{N} P_1 r k}$$

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

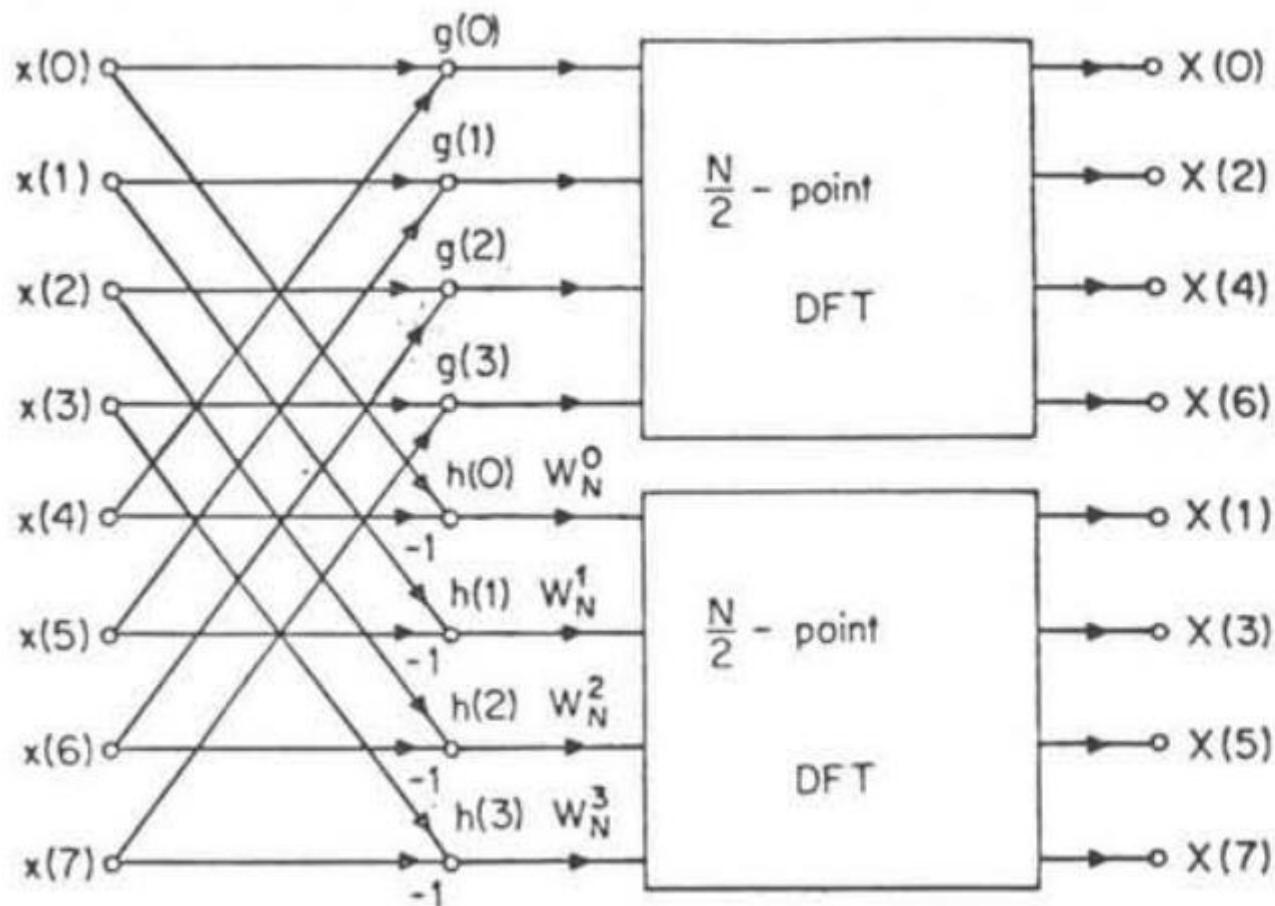
$$X(k) = \sum_{l=0}^{P_1-1} W_N^{lk} \underbrace{\sum_{r=0}^{q_1-1} \chi(P_1 r + l) W_{q_1}^{rk}}_{q_1\text{-point DFT}}$$

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

MADS  $\Rightarrow$

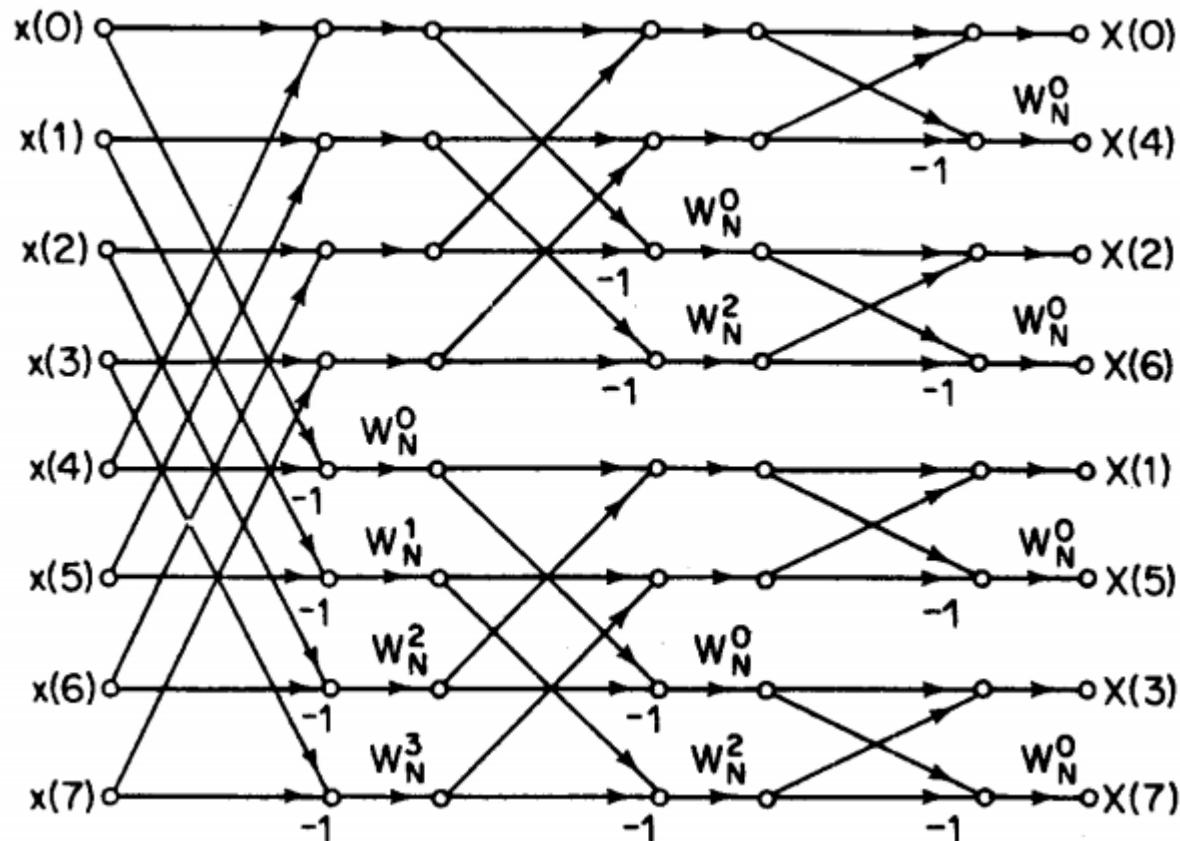
$$N[P_1 + P_2 + \cdots + P_v - v]$$

# Class Review



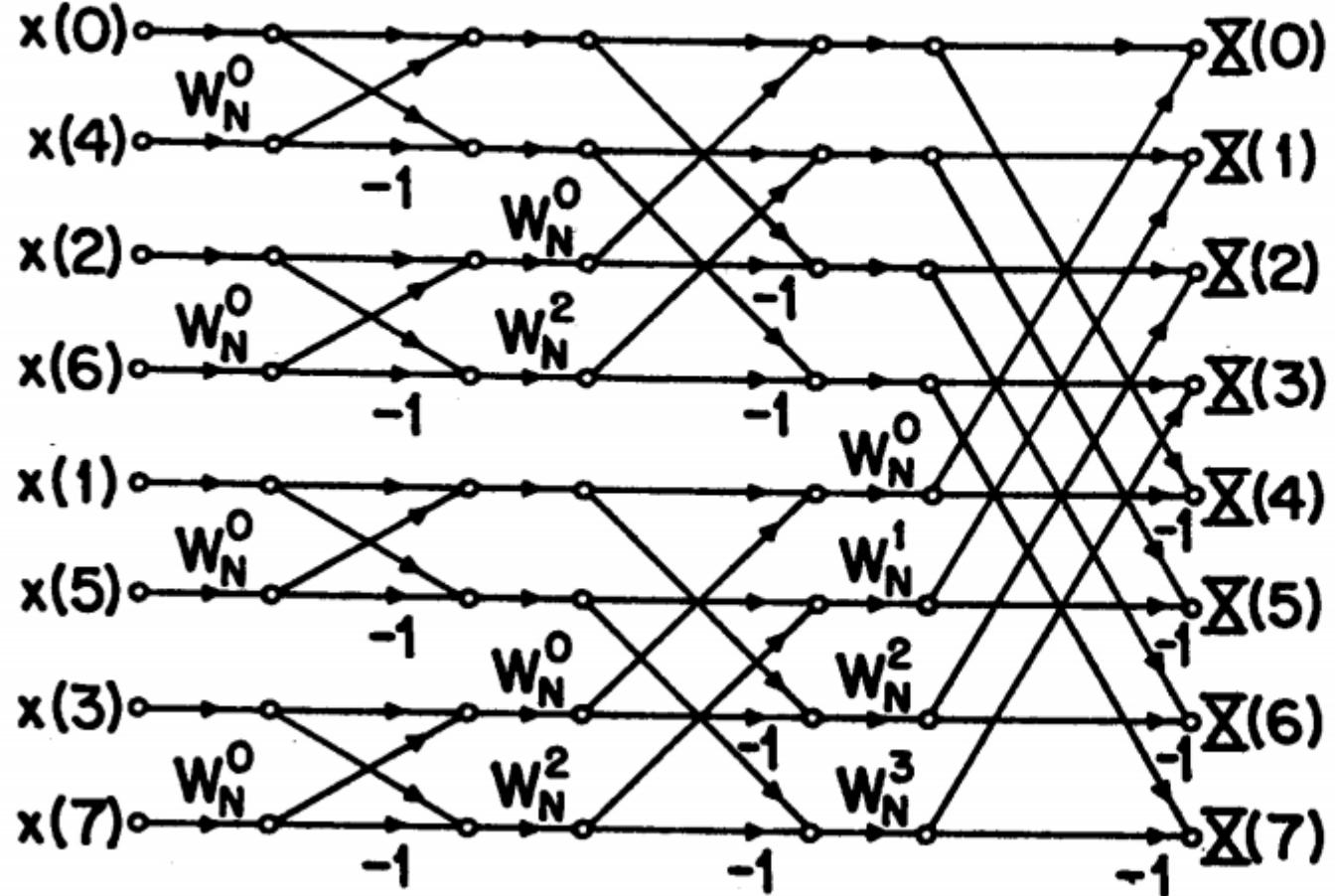
Computation of even and odd numbered DFT values.

# Class Review



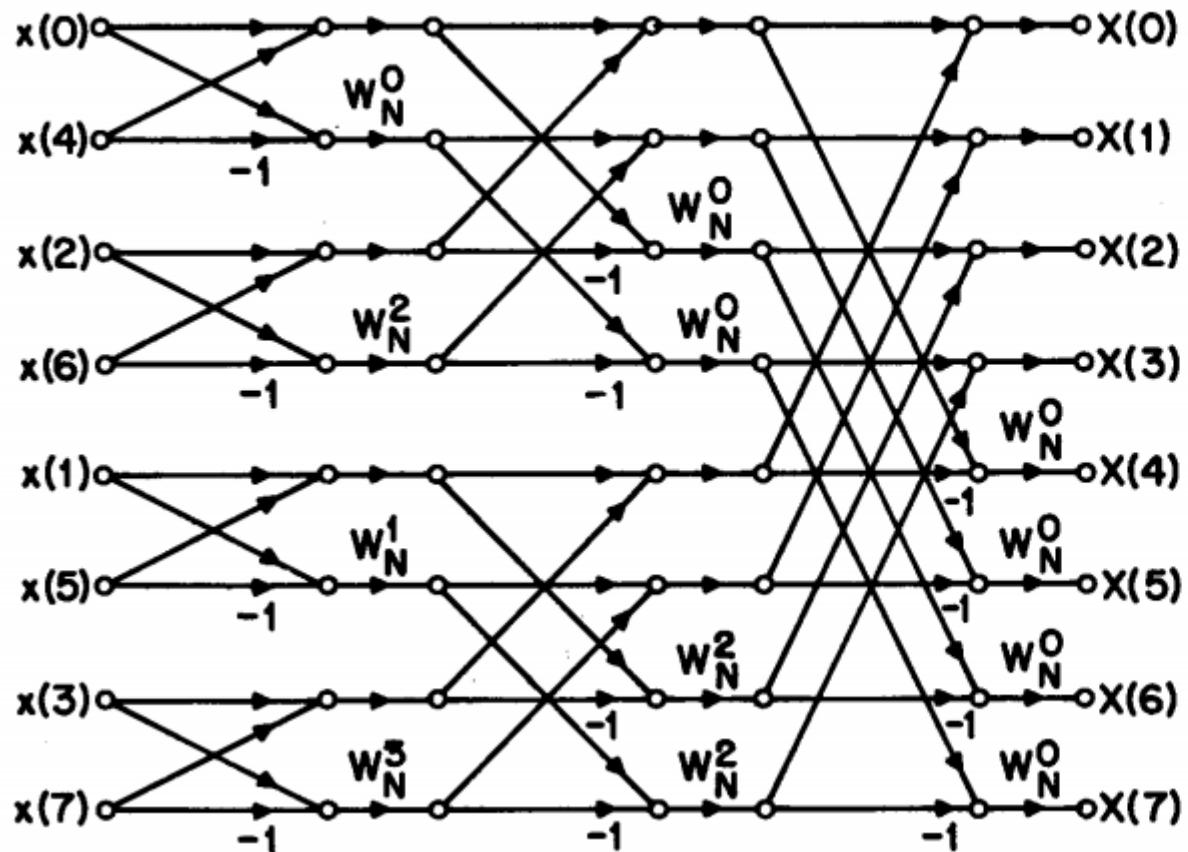
**Flow-graph of complete decimation-in-frequency decomposition of an eight point DFT computation.**

# Class Review



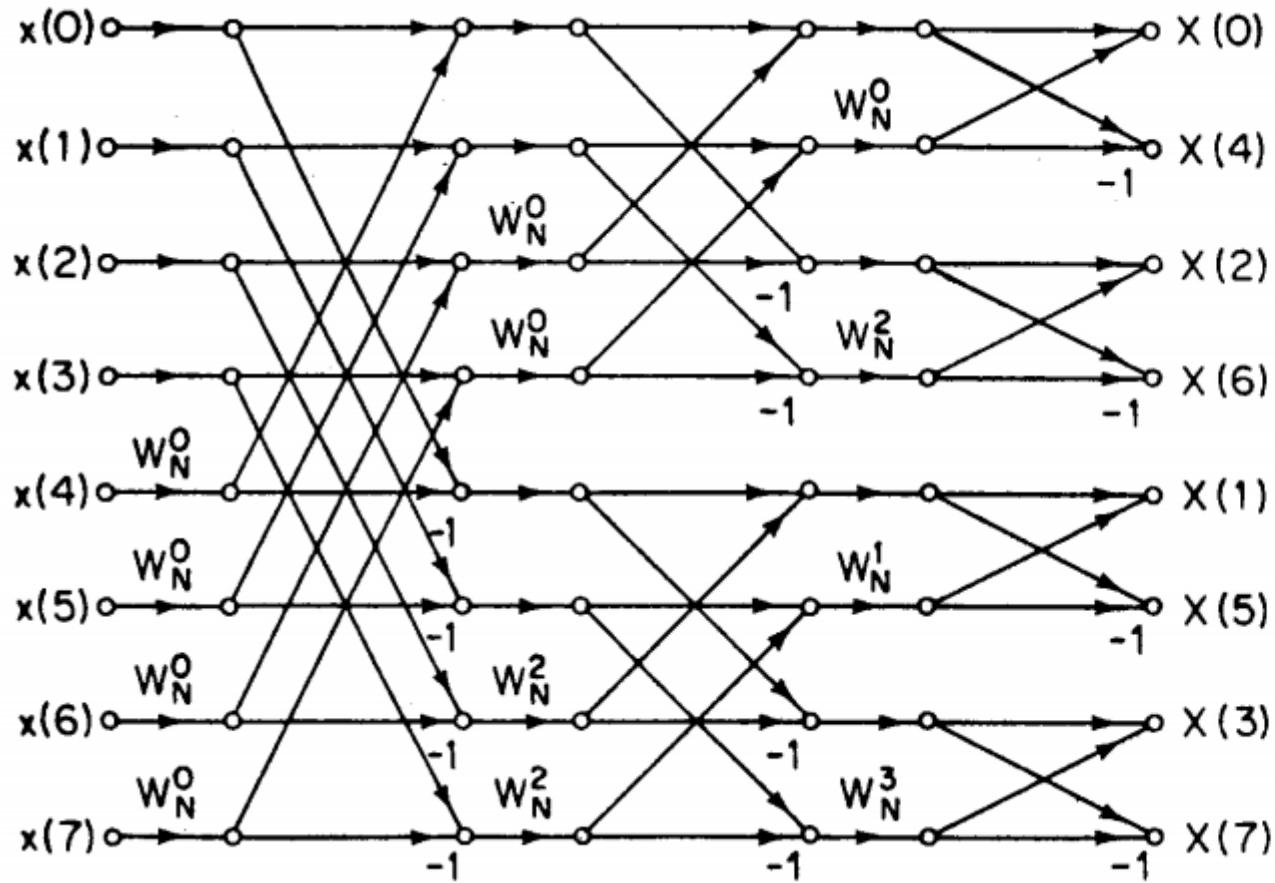
FFT flow-graph for decimation-in-time algorithm.

# Class Review



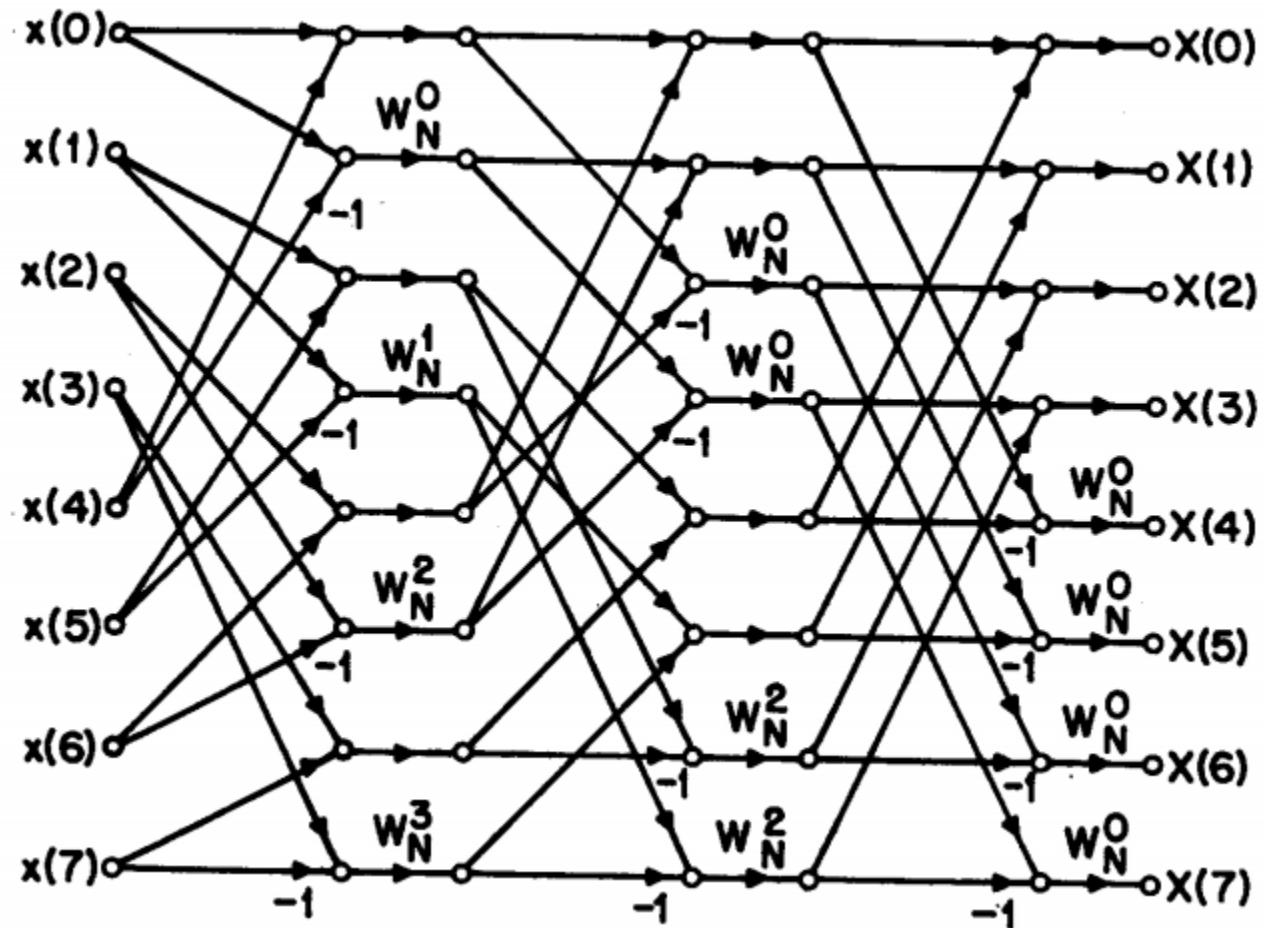
Rearrangement of the decimation-in-frequency flow-graph d. The input is now in bit-reversed order and the output is in normal order.

# Class Review



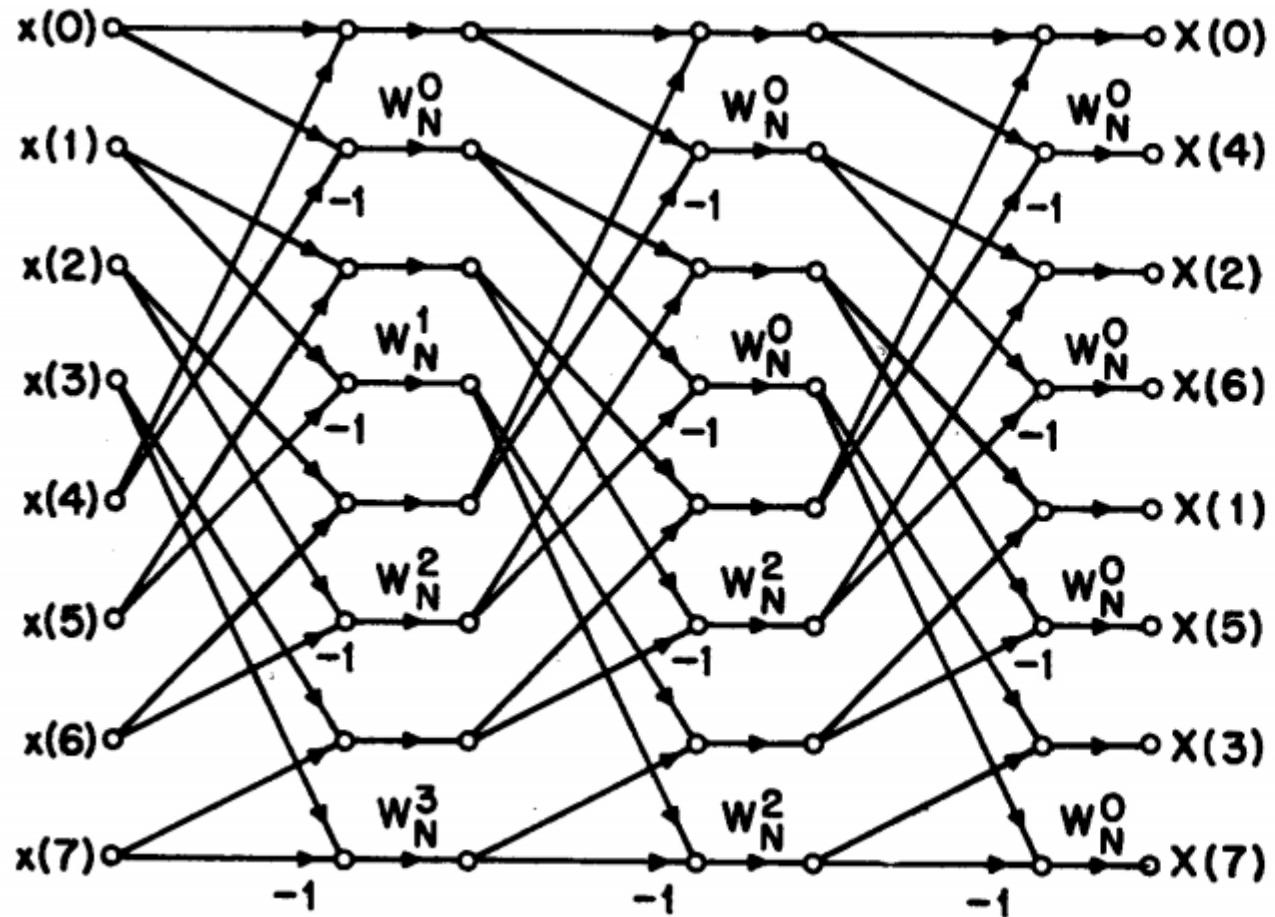
Rearrangement of e so that the input is in normal order and output in bit-reversed order.

# Class Review



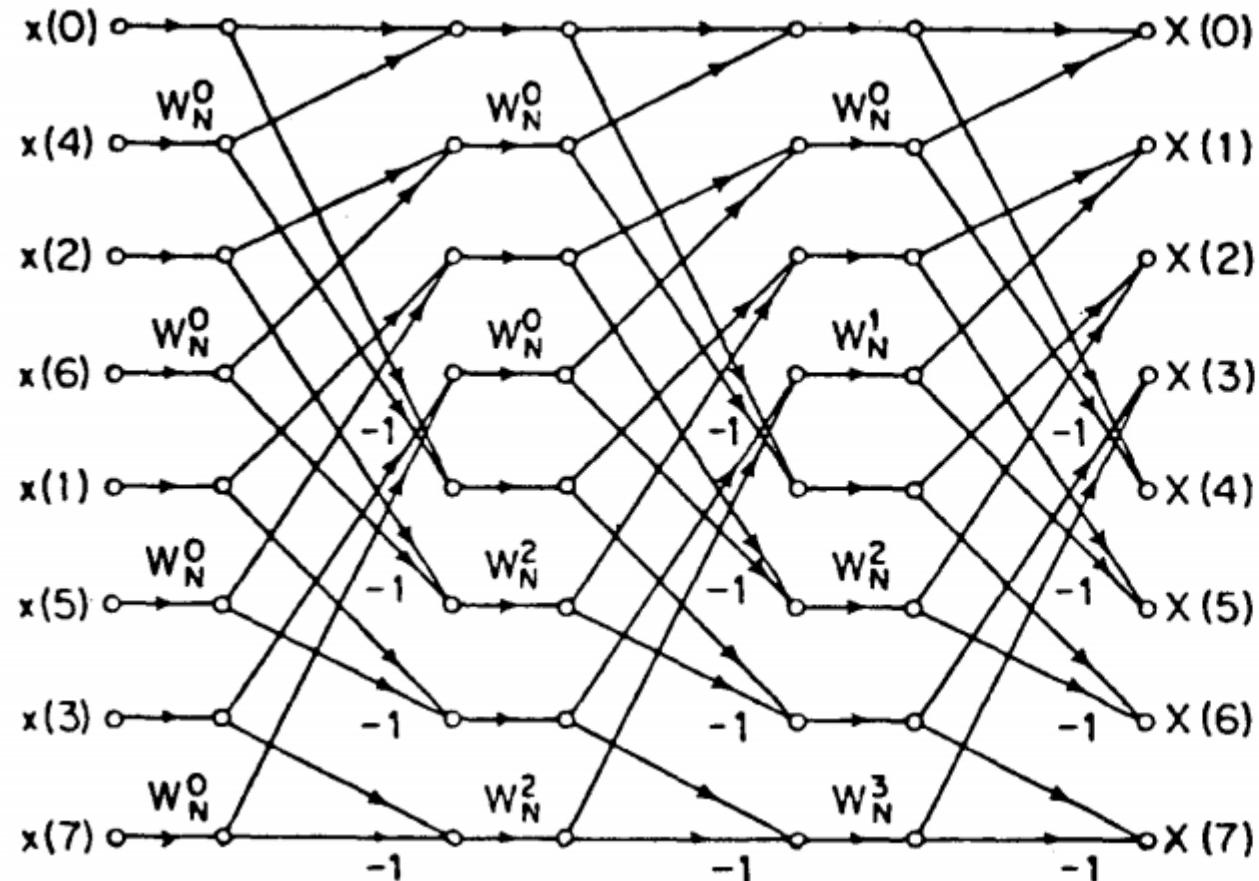
Rearrangement of  $d$  so that both input and output are in normal order.

# Class Review



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

# Class Review



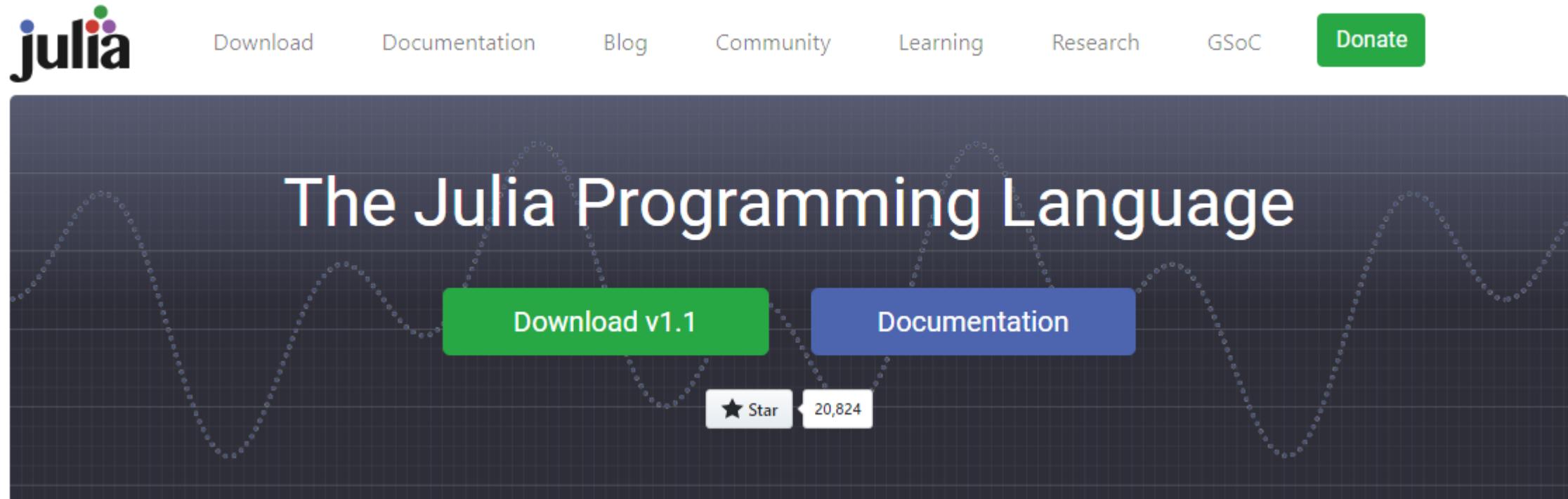
Decimation-in-time flow-graph for which the geometry is identical for each stage.

# Optimization

Convex Optimization

**Convex.jl** , cvx, cvxpy, cvxr

# Julia Programming Language for Optimization

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad i = 1, \dots, p, \end{aligned}$$


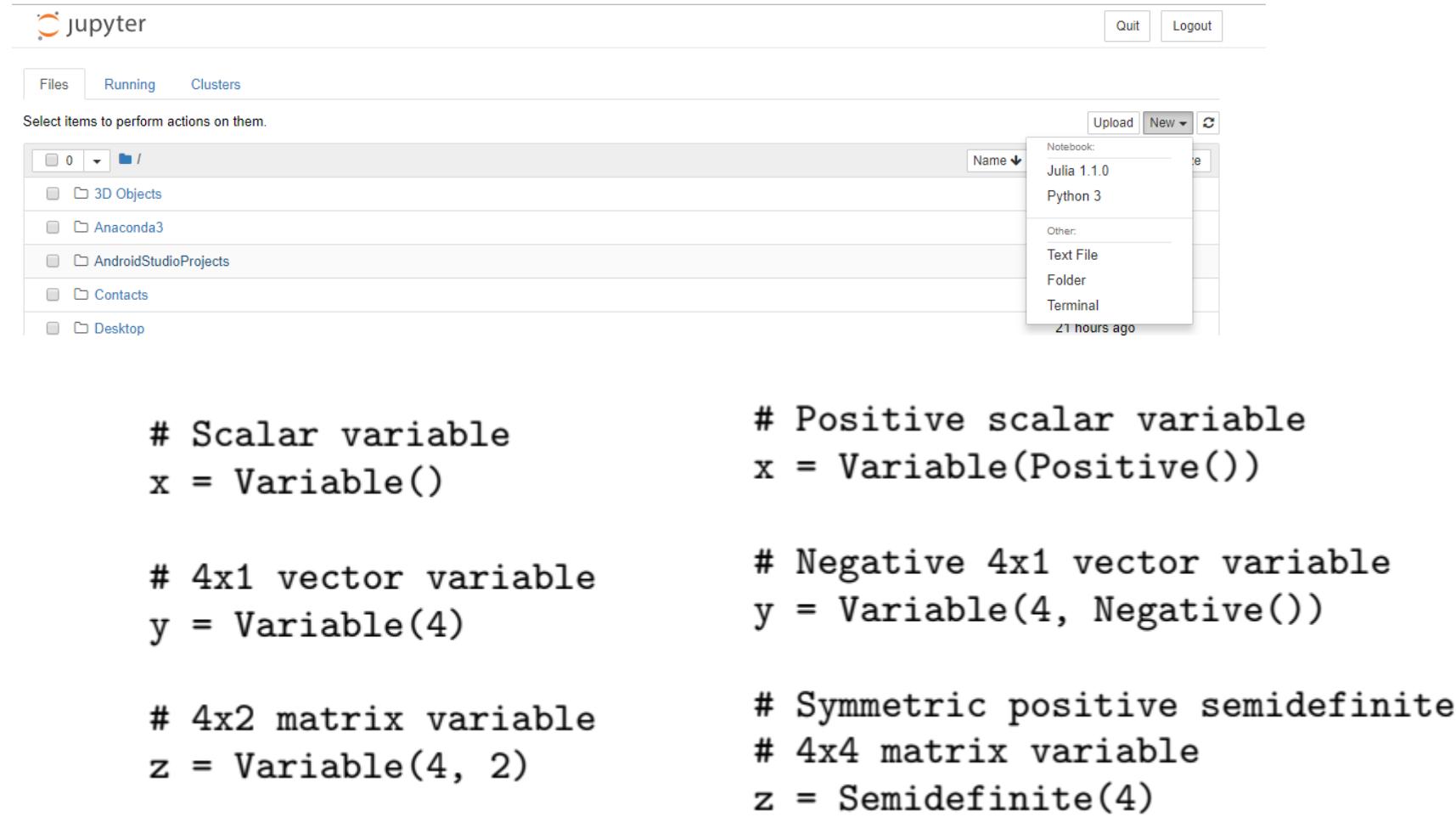
The screenshot shows the official website for the Julia programming language. The top navigation bar includes links for Download, Documentation, Blog, Community, Learning, Research, GSoC, and a green 'Donate' button. The main banner features the text 'The Julia Programming Language' over a background of a mathematical plot with oscillating data points. Below the banner are two prominent buttons: a green 'Download v1.1' button and a blue 'Documentation' button. At the bottom of the banner, there is a 'Star' icon with the number '20,824'.

# Julia Programming Language for Optimization

```
Import Pkg
Pkg.add("IJulia")
```

```
Import Pkg
Pkg.add("SCS")
Pkg.add("Convex")
```

```
using Convex
x=Variable()
p=minimize(x,x>=0)
solve(!p)
```



The screenshot shows a Jupyter Notebook interface with the title "jupyter". The top menu bar includes "Files", "Running", "Clusters", "Quit", and "Logout". A sidebar on the left lists directories: "0", "3D Objects", "Anaconda3", "AndroidStudioProjects", "Contacts", and "Desktop". A dropdown menu on the right is open, showing options: "Name", "Upload", "New", "Notebook: Julia 1.1.0", "Python 3", "Other: Text File, Folder, Terminal", and a timestamp "21 hours ago". The main area contains the following Julia code examples:

```

# Scalar variable
x = Variable()

# 4x1 vector variable
y = Variable(4)

# 4x2 matrix variable
z = Variable(4, 2)

# Positive scalar variable
x = Variable(Positive())

# Negative 4x1 vector variable
y = Variable(4, Negative())

# Symmetric positive semidefinite
# 4x4 matrix variable
z = Semidefinite(4)

```

# Julia Programming Language for Optimization

```

# indexing, multiplication, addition
e1 = y[1] + 2*x

# expressions can be affine, convex, or concave
e3 = sqrt(x) + log(x)

# more atoms
e2 = 4 * pos(x) + max(abs(y)) + norm(z[,1],2)

x = Variable()
e = max(x,0)
x.value = -4
evaluate!(e)

```

# Julia Programming Language for Optimization

```
# affine equality constraint
A = randn(3,4); b = randn(3)
c1 = A*y == b

# convex inequality constraint
c2 = norm(y,2) <= 2
```

minimize  $\|x\|_\infty$   
 subject to  $x_1 + x_2 = 5$   
 $x_3 \leq x_2$ ,

```
x = Variable(3)
constraints = [x[1]+x[2] == 5, x[3] <= x[2]]
p = minimize(norm_inf(x), constraints)
```

# Julia Programming Language for Optimization

## Linear program

$$\begin{aligned}
 & \text{maximize} && c^T x \\
 & \text{subject to} && Ax \leq b \\
 & && x \geq 1 \\
 & && x \leq 10 \\
 & && x_2 \leq 5 \\
 & && x_1 + x_4 - x_2 \leq 10
 \end{aligned}$$

```

x = Variable(4)
c = [1; 2; 3; 4]
A = eye(4)
b = [10; 10; 10; 10]
p = minimize(dot(c, x)) # or c' * x
p.constraints += A * x <= b
p.constraints += [x >= 1; x <= 10; x[2] <= 5; x[1] + x[4] - x[2] <= 10]
solve!(p)

println(round(p.optval, 2))
println(round(x.value, 2))
println(evaluate(x[1] + x[4] - x[2]))

```

# Julia Programming Language for Optimization

## Matrix Variables and promotions

$$\begin{aligned} \text{minimize} \quad & \|X\|_F + y \\ \text{subject to} \quad & 2X \leq 1 \\ & X' + y \geq 1 \\ & X \geq 0 \\ & y \geq 0 \end{aligned}$$

```
X = Variable(2, 2)
y = Variable()
# X is a 2 x 2 variable, and y is scalar. X' + y promotes y to a 2 x 2 variable before adding them
p = minimize(vecnorm(X) + y, 2 * X <= 1, X' + y >= 1, X >= 0, y >= 0)
solve!(p)
println(round(X.value, 2))
println(y.value)
p.optval
```

# Julia Programming Language for Optimization

## Norm, exponential and geometric mean

$$\text{satisfy } \|x\|_2 \leq 100$$

$$e^{x_1} \leq 5$$

$$x_2 \geq 7$$

$$\sqrt{x_3 x_4} \geq x_2$$

```
x = Variable(4)
p = satisfy(norm(x) <= 100, exp(x[1]) <= 5, x[2] >= 7, geomean(x[3], x[4]) >= x[2])
solve!(p, SCSSolver(verbose=0))
println(p.status)
x.value
```

# Julia Programming Language for Optimization

## SDP cone and Eigenvalues

```
y = Semidefinite(2)
p = maximize(lambdamin(y), trace(y)<=6)
solve!(p, SCSSolver(verbose=0))
p.optval
```

```
(size(coeff),size(var)) = ((2,2),(2,2))
(size(coeff),size(var)) = ((2,2),(2,2))
```

```
x = Variable()
y = Variable((2, 2))
# SDP constraints
p = minimize(x + y[1, 1], isposdef(y), x >= 1, y[2, 1] == 1)
solve!(p)
y.value
```

# Julia Programming Language for Optimization

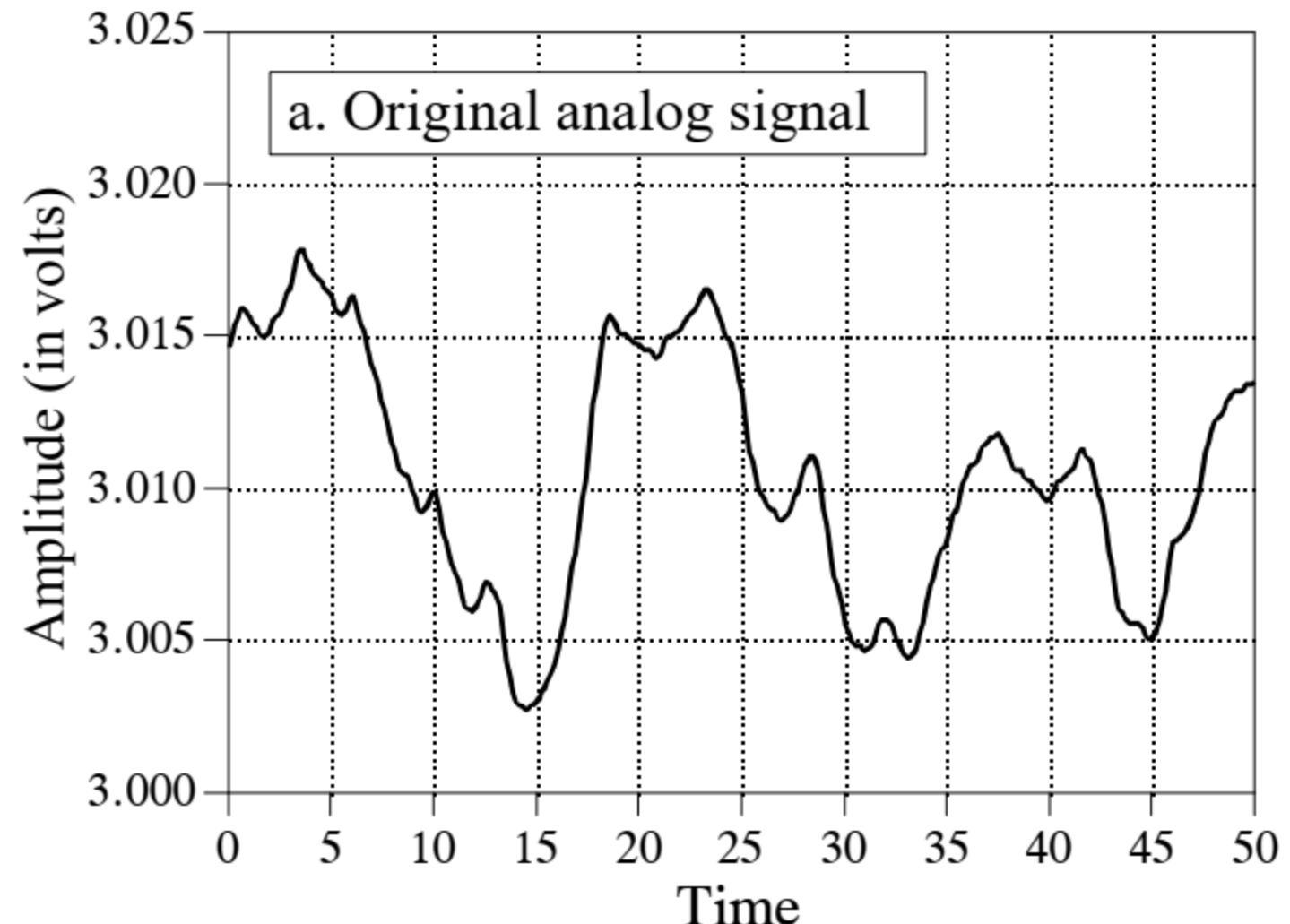
## Mixed integer program

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n x_i \\ & \text{subject to} && x \in \mathbb{Z}^n \\ & && x \geq 0.5 \end{aligned}$$

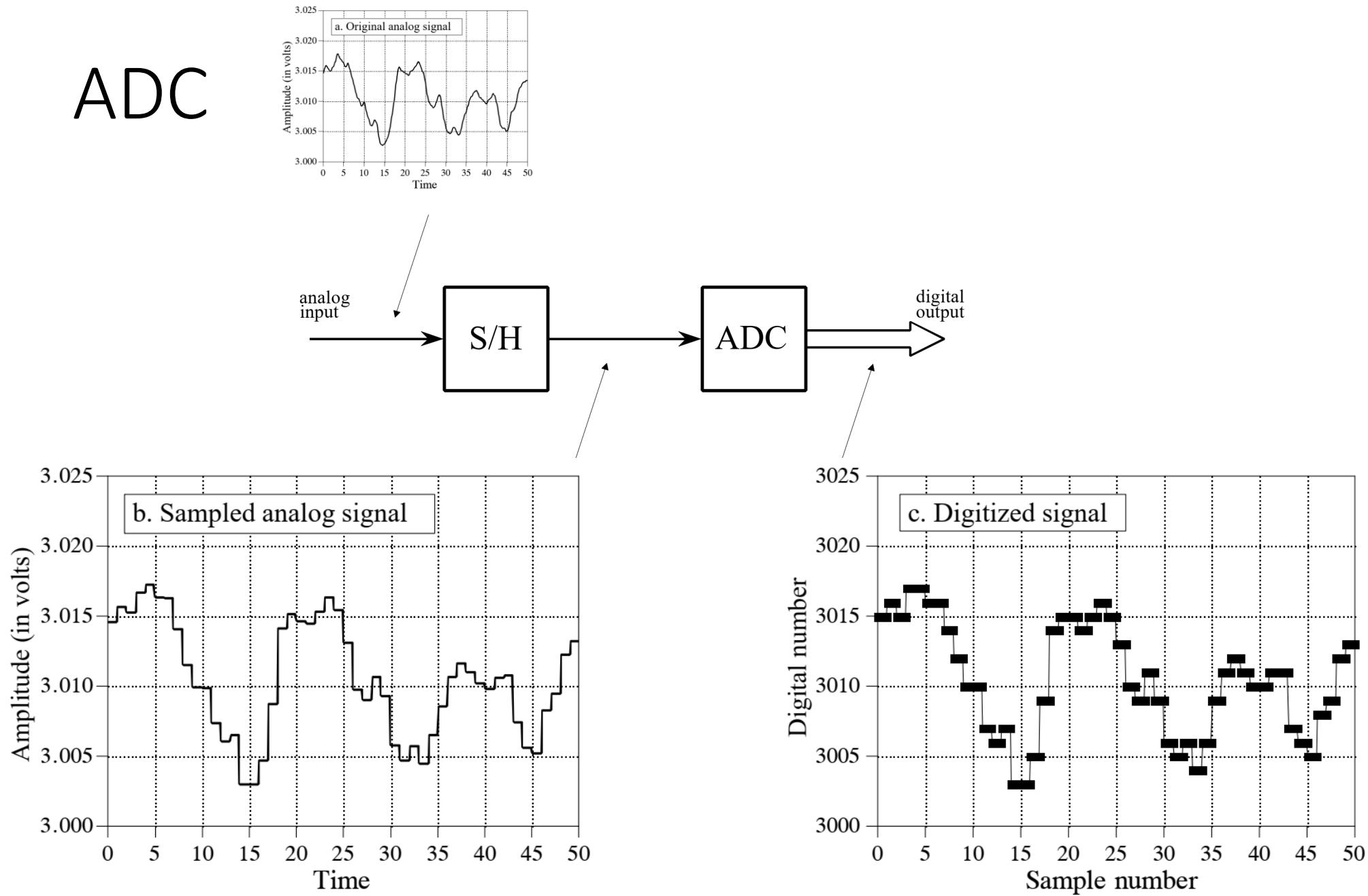
```
using GLPKMathProgInterface
x = Variable(4, :Int)
p = minimize(sum(x), x >= 0.5)
solve!(p, GLPKSolverMIP())
x.value
```

# ADC

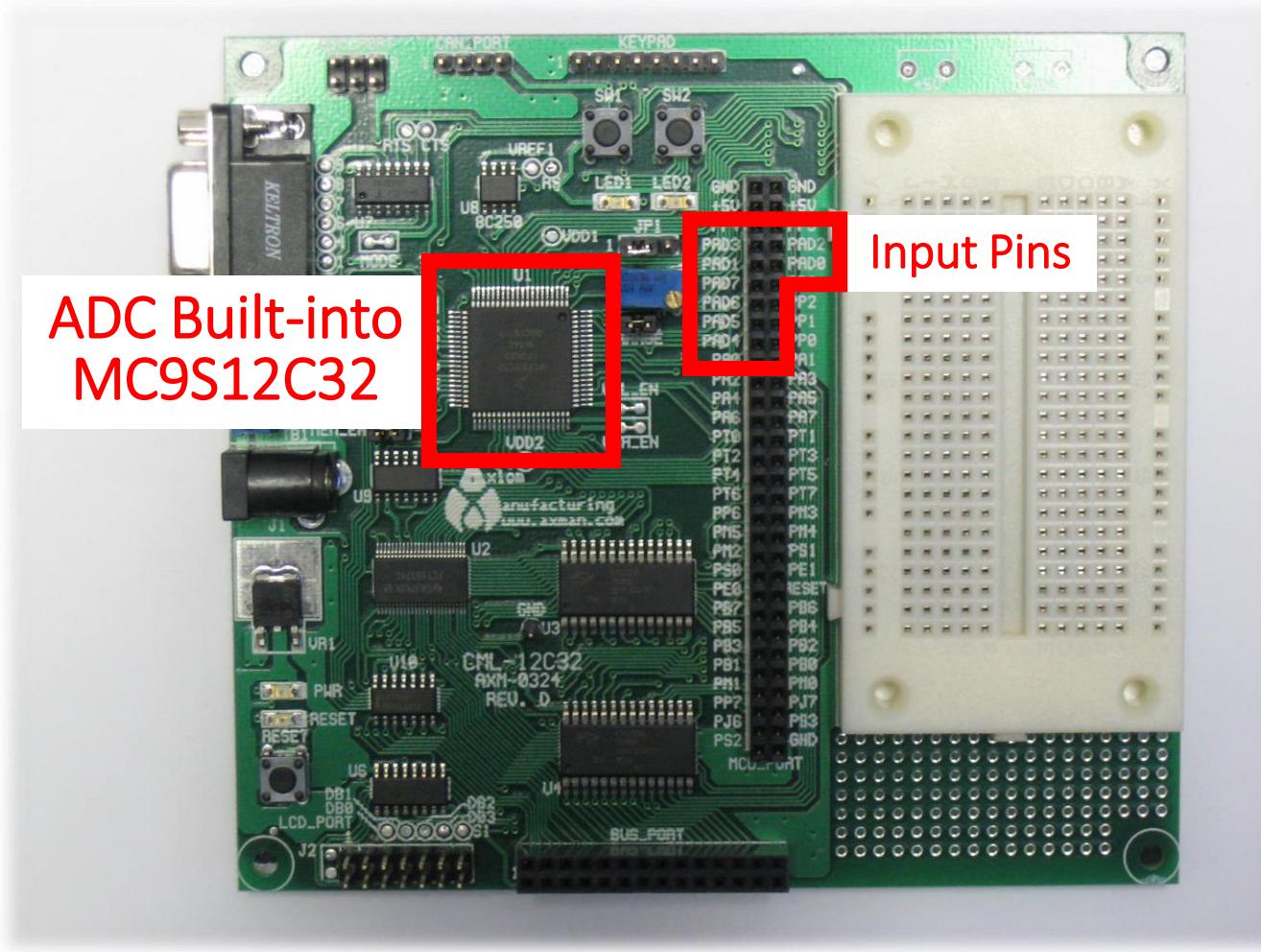
# ADC



# ADC



# ADC

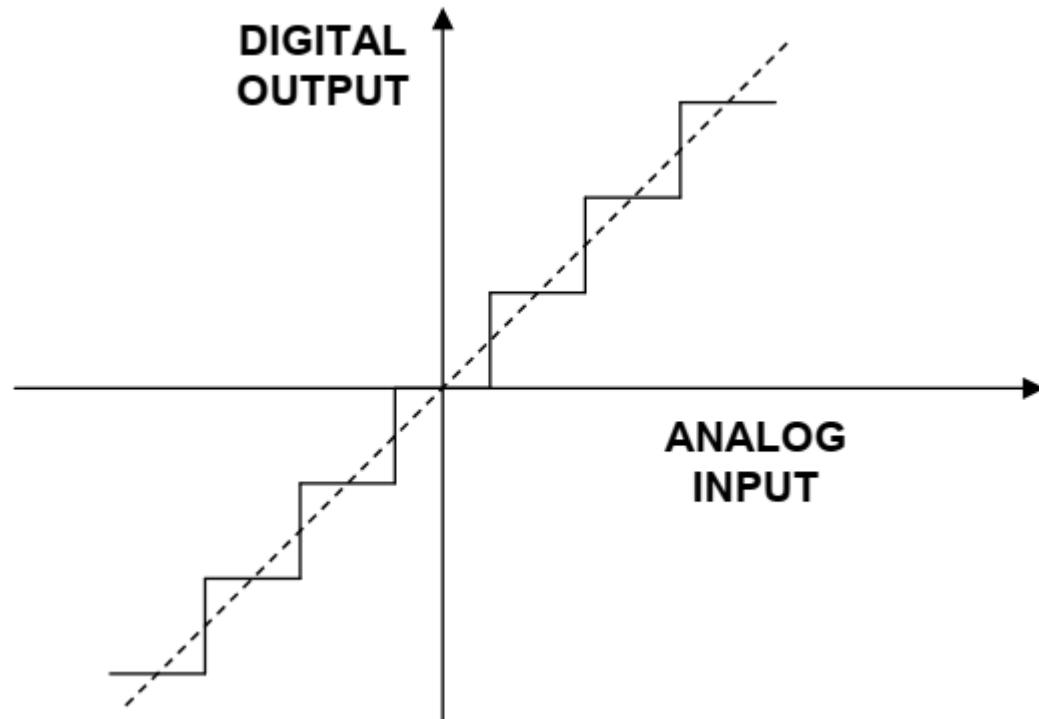


# ADC

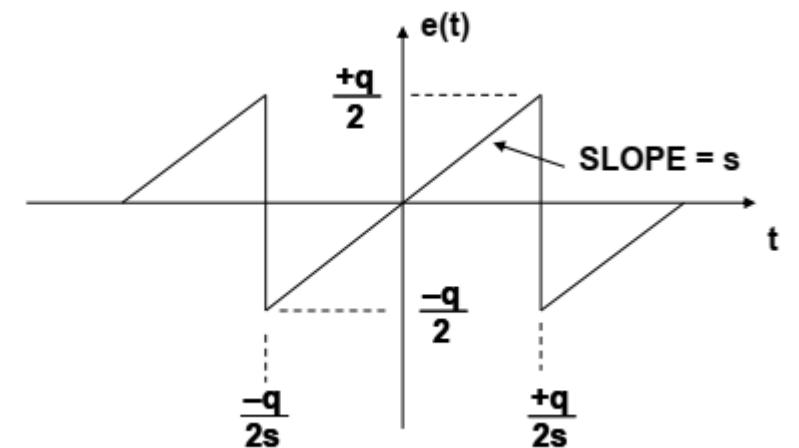
ADC Sampling Rate (500MHz)

ADC Quantization Bits (12 bit)

# ADC Quantization Noise



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



# ADC Quantization Noise

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

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

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

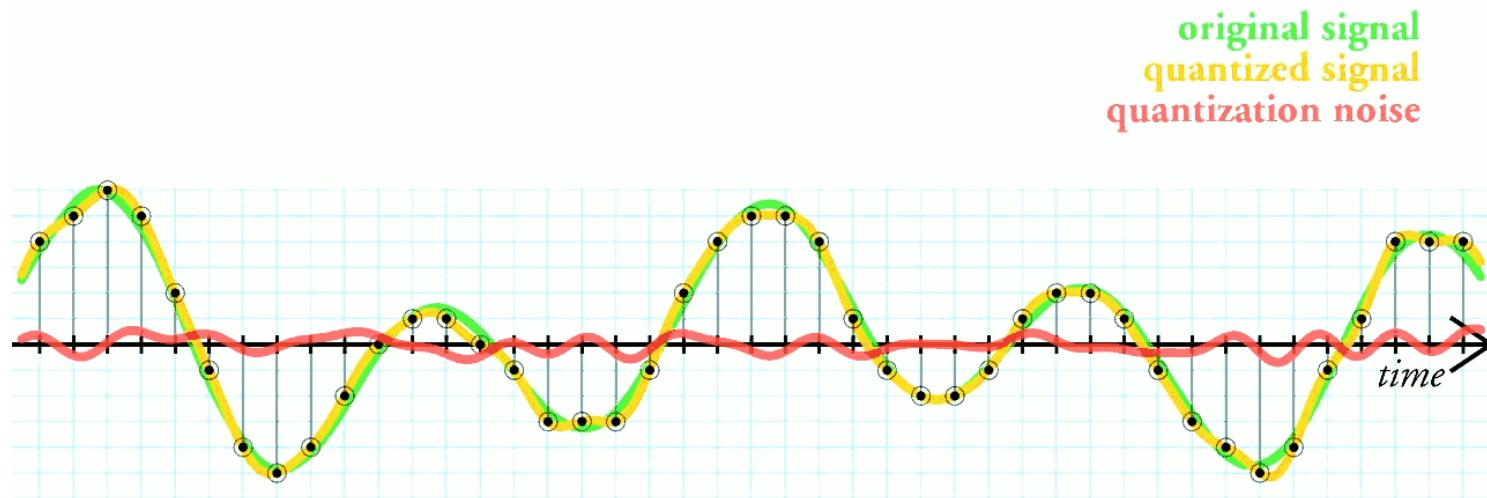
# ADC Quantization Noise

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

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

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

# 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:  $v(t) = \frac{q2^N}{2} \sin(2\pi ft)$ .

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

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

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

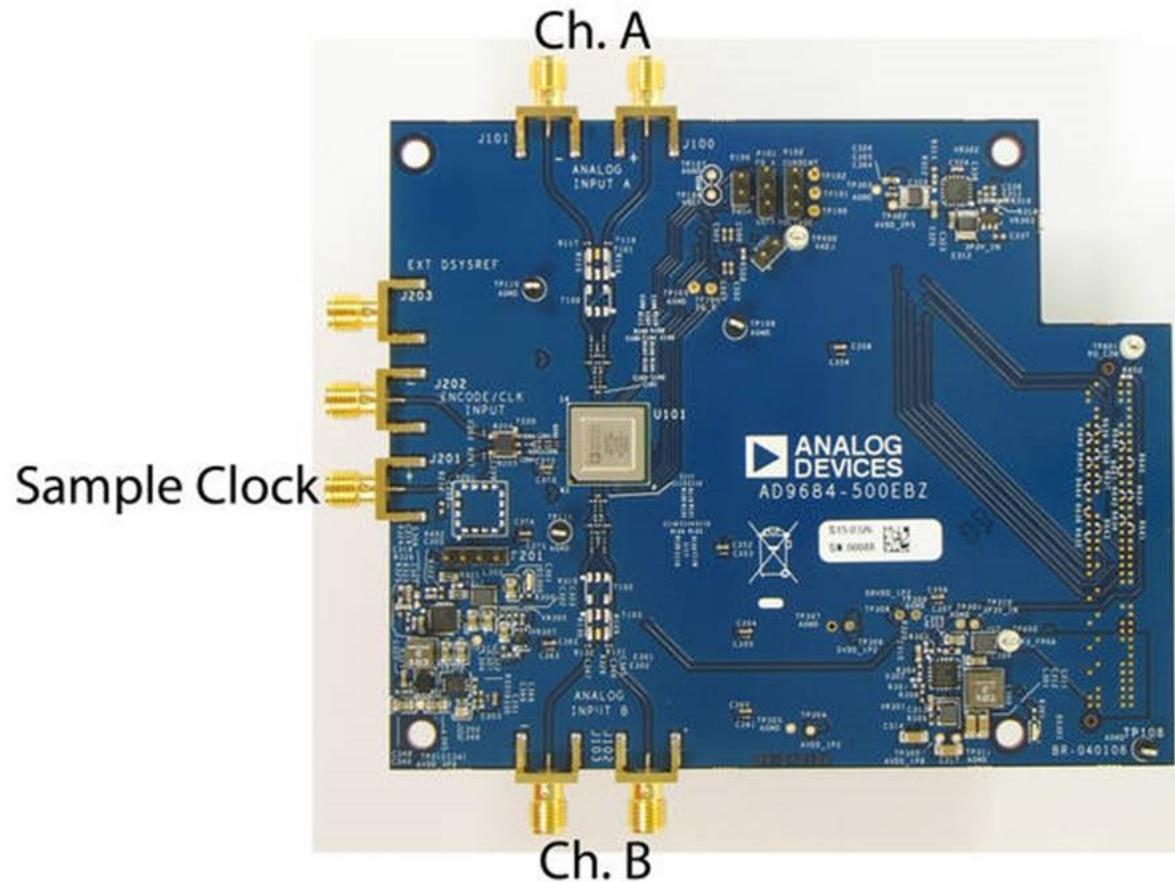
# ADC Quantization SNR (N Bit)

$$\text{SNR} = 6.02N + 1.76\text{dB}$$

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

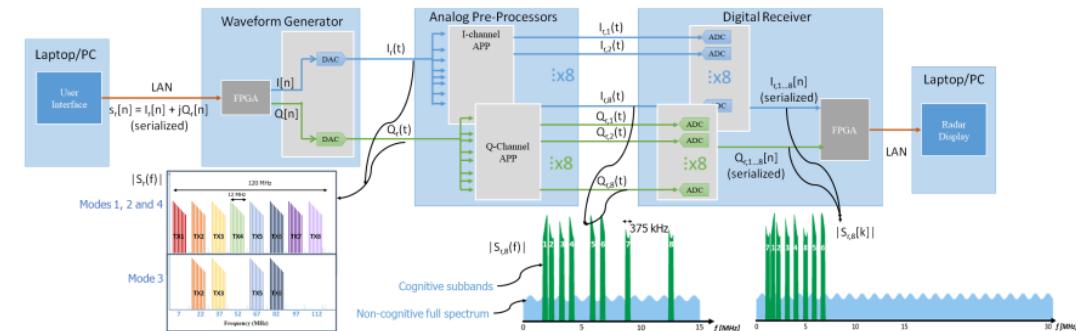
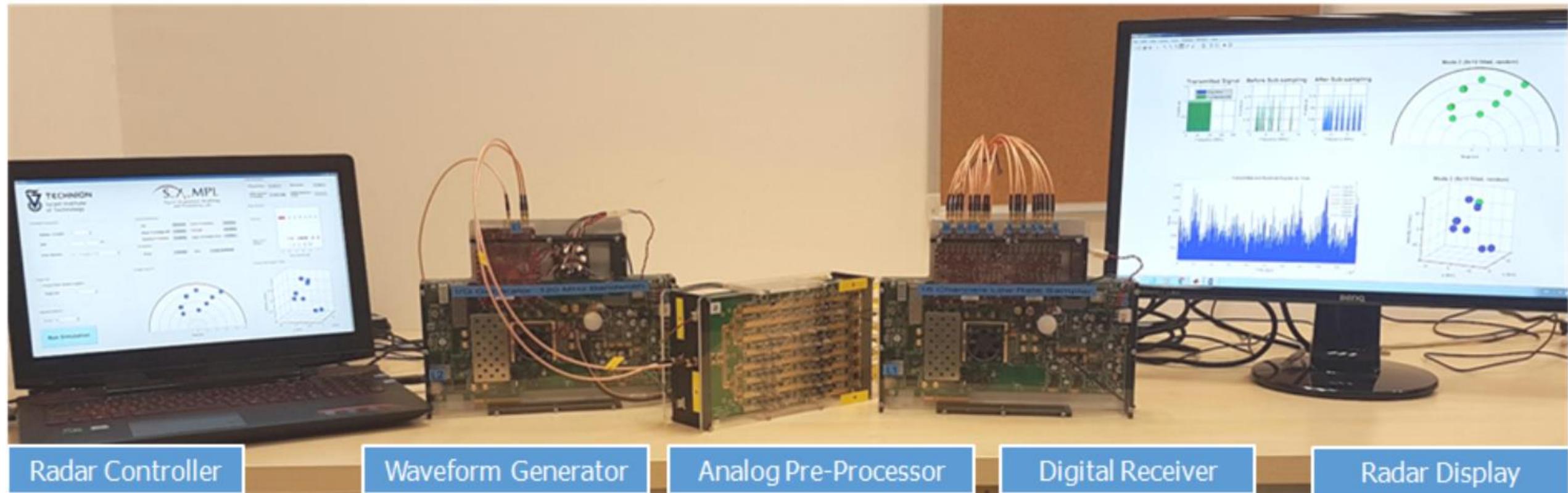
# ADC

# AD9684



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

# ADC in SUMMeR (Sub-Nyquist colocated MiMo Radar)



ADC

**AD9684**



**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 =  $\pm 0.5$  LSB

INL =  $\pm 2.5$  LSB

Noise density = -153 dBFS/Hz at 500 MSPS

1.25 V, 2.50 V, and 3.3 V supply operation

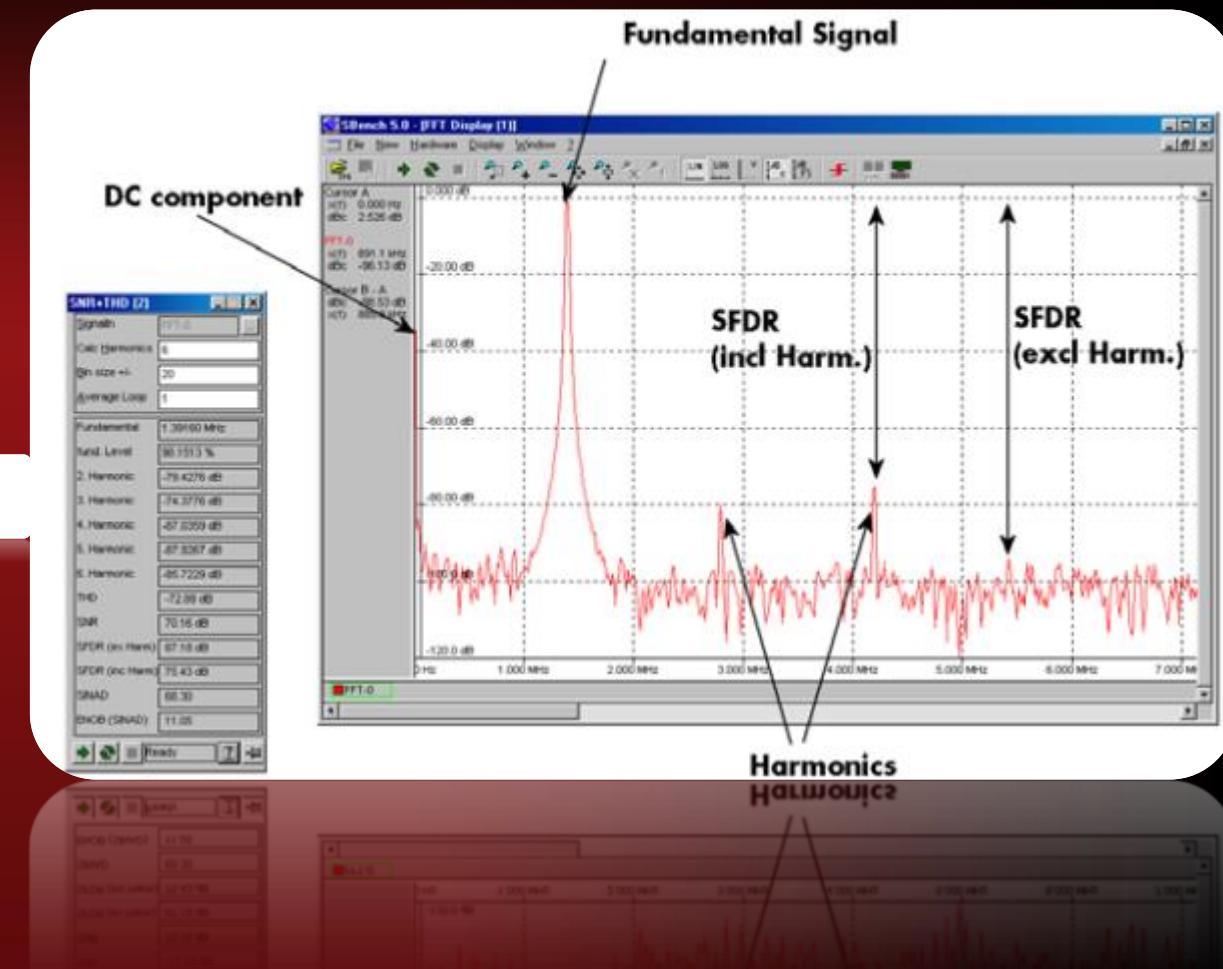
# ADC

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

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

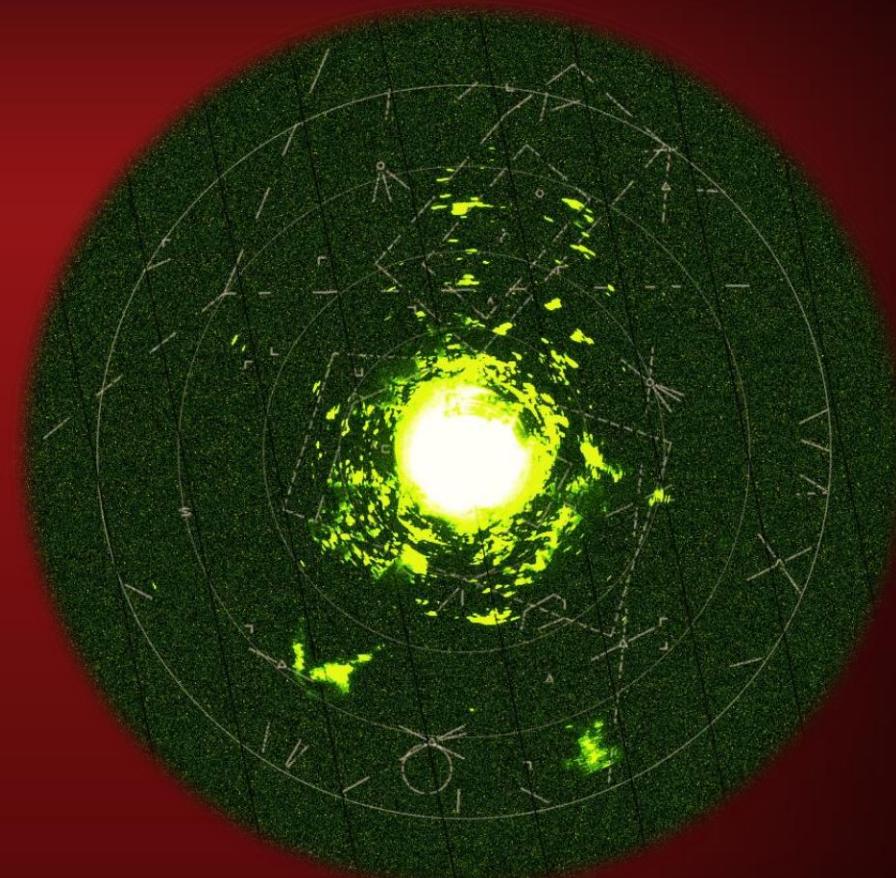
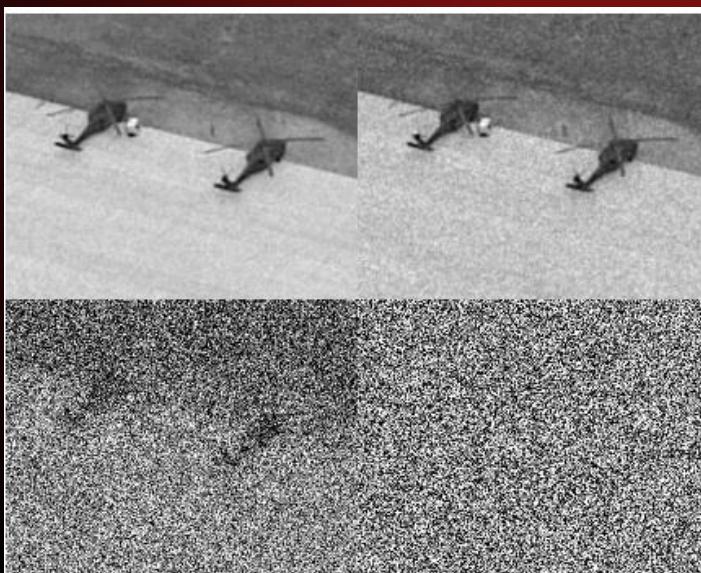
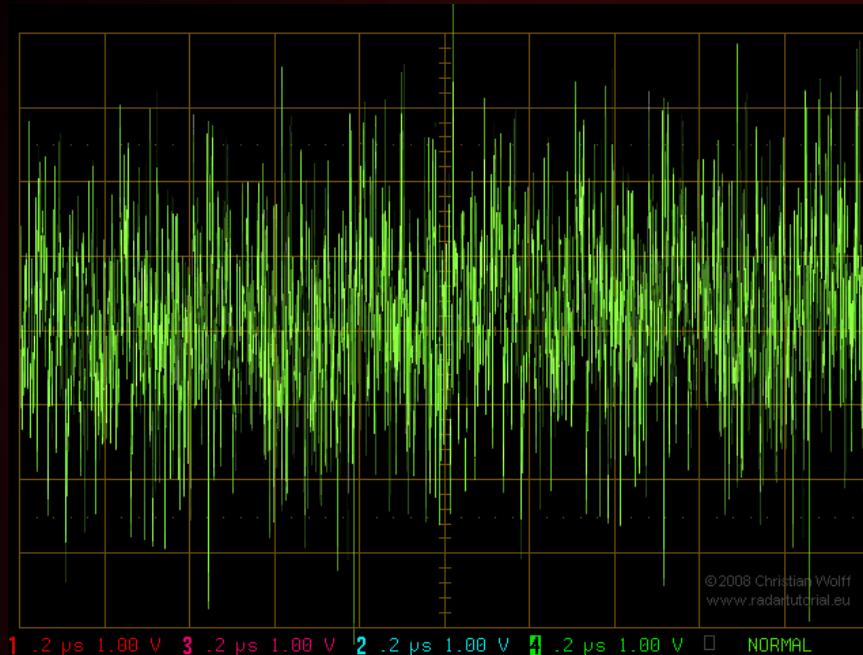
$$SINAD = 20 * \log ([\text{Fundamental}] / \text{SQRT} (\text{SUM} (\text{SQR}([\text{Noise + Harmonics}])))$$

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

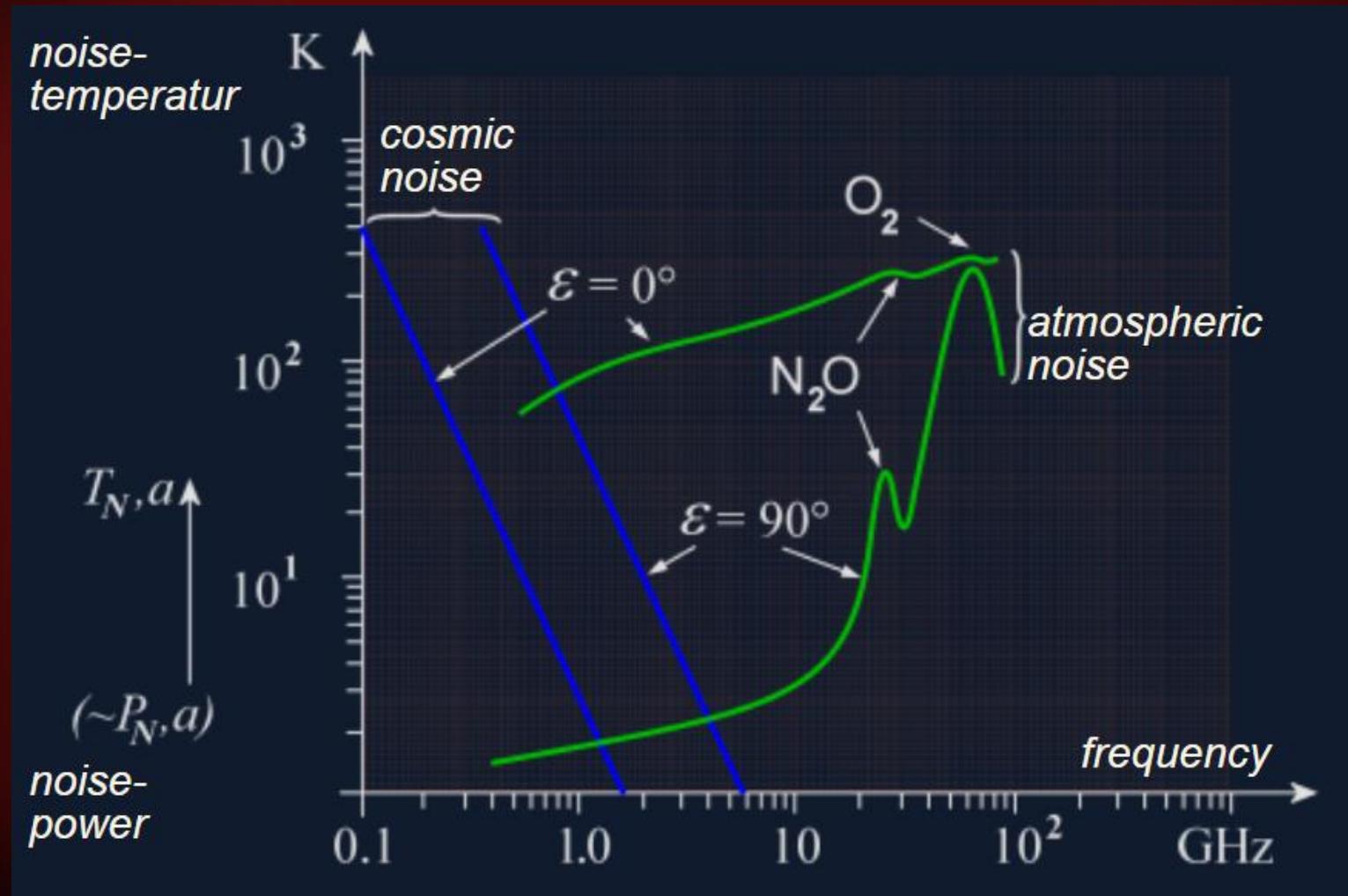


# Noise

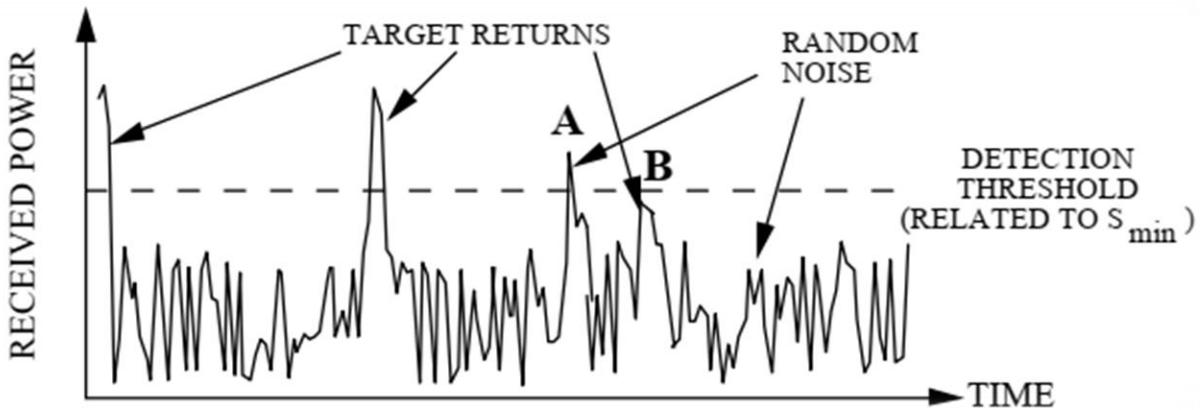
# Noise



# Noise



# 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

$$N_o = kT_o B_n$$

where  $k_B = 1.38 \times 10^{-23}$  (Joules/K) is Boltzman's constant.

$$\text{SNR} = \frac{P_r}{N_o} = \frac{P_t G_t G_r \sigma \lambda^2 G_p L}{(4\pi)^3 R^4 k_B T_s B_n}$$

# Thermal Noise

$$N_o = kT_o B_n$$

```
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))
```

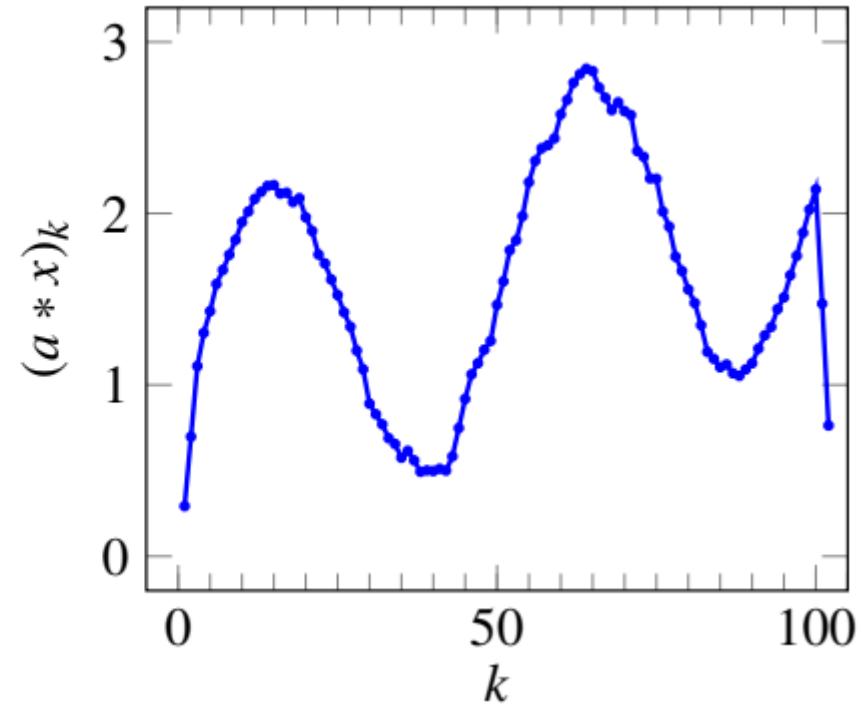
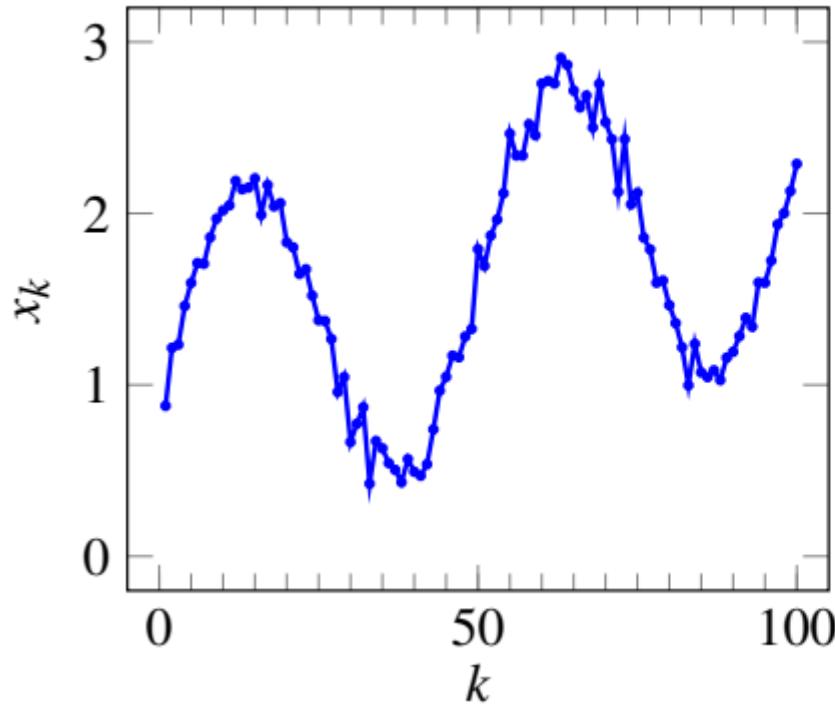
```
Noise Power for BW: 1 KHz = -173.97722915699808 dB = -143.97722915699808 dBm
```

# System Noise

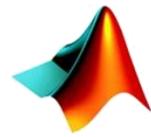
$$\text{Noise floor}_{\text{dBm}} = 10 \log_{10}(k \times T_0 \times 1000) + \text{NF} + 10 \log_{10} \text{BW}$$

# Moving Average Filter

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



# Python Numpy & MATLAB



MATLAB®



## MATLAB

help func

## numpy

info(func) or help(func) or func? (in Ipython)

which func

see note HELP

type func

source(func) or func?? (in Ipython)

a &amp;&amp; b

a and b

a || b

a or b

1\*i , 1\*j , 1i , 1j

1j

eps

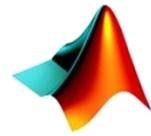
np.spacing(1)

ode45

scipy.integrate.solve\_ivp(f)

ode15s

scipy.integrate.solve\_ivp(f, method='BDF')



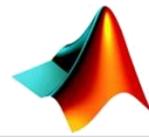
MATLAB®



python™

**MATLAB**
**NumPy**

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

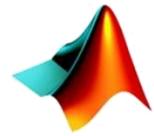


MATLAB®



python™

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

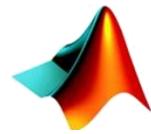


MATLAB®



python™

<code>a .* (a&gt;0.5)</code>	<code>a * (a&gt;0.5)</code>
<code>a(:) = 3</code>	<code>a[:] = 3</code>
<code>y=x</code>	<code>y = x.copy()</code>
<code>y=x(2,:)</code>	<code>y = x[1,:].copy()</code>
<code>y=x(:)</code>	<code>y = x.flatten()</code>
<code>1:10</code>	<code>arange(1.,11.) or r_[1.:11.] or r_[1:10:10j]</code>
<code>0:9</code>	<code>arange(10.) or r_[:10.] or r_[:9:10j]</code>
<code>[1:10]'</code>	<code>arange(1.,11.)[ :, newaxis ]</code>
<code>zeros(3,4)</code>	<code>zeros((3,4))</code>
<code>zeros(3,4,5)</code>	<code>zeros((3,4,5))</code>
<code>ones(3,4)</code>	<code>ones((3,4))</code>
<code>eye(3)</code>	<code>eye(3)</code>
<code>diag(a)</code>	<code>diag(a)</code>
<code>diag(a,0)</code>	<code>diag(a,0)</code>

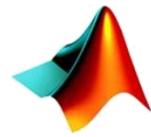


MATLAB®



python™

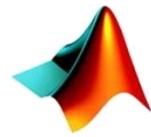
<code>rand(3,4)</code>	<code>random.rand(3,4)</code>
<code>linspace(1,3,4)</code>	<code>linspace(1,3,4)</code>
<code>[x,y]=meshgrid(0:8,0:5)</code>	<code>mgrid[0:9.,0:6.] or meshgrid(r_[0:9.],r_[0:6.])</code>
	<code>ogrid[0:9.,0:6.] or ix_(r_[0:9.],r_[0:6.])</code>
<code>[x,y]=meshgrid([1,2,4],[2,4,5])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



MATLAB®



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



MATLAB®



python™

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

# Extra Programming Skills

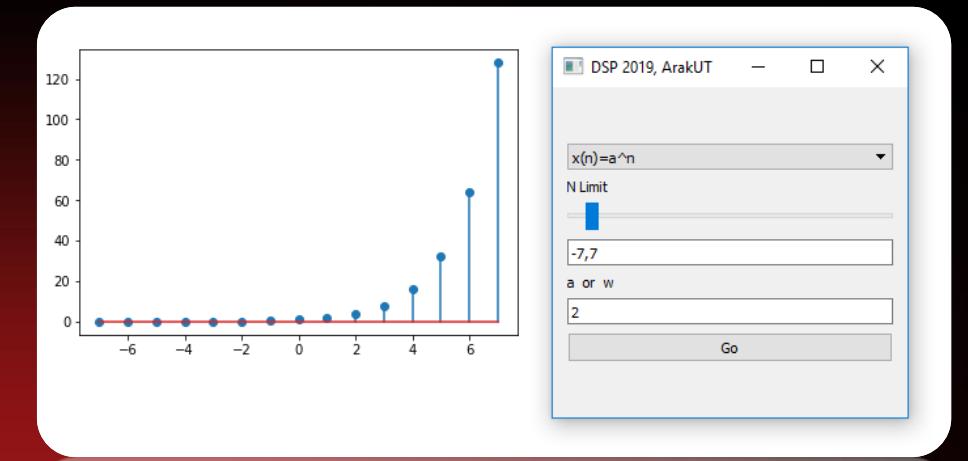
# DSP+Python UI

```
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(dspEdita.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)
```



SPYDER



```
pb.clicked.connect(pbslot)
dspLineEdit=QLineEdit("-1,1",frame)
dspEdita=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(dspEdita)
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 = null  
~MainWindow();  
    QTcpServer *server;  
    QTcpSocket *socket;  
private:  
    Ui::MainWindow *ui;  
public slots:  
    void newConnection();  
    void readyRead();  
private slots:  
    void on_pushButton_clicked();  
    void on_pushButton_2_clicked();  
private slots:  
    void readyRead();
```

# TCP Connection in Python and C++

```
import socket
TCP_IP = '127.0.0.1'
TCP_PORT = 5462
BUFFER_SIZE = 1024
MESSAGE = "Hello, World!"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE.encode('utf-8'))
data = s.recv(BUFFER_SIZE)
s.close()
print("received data:", data)

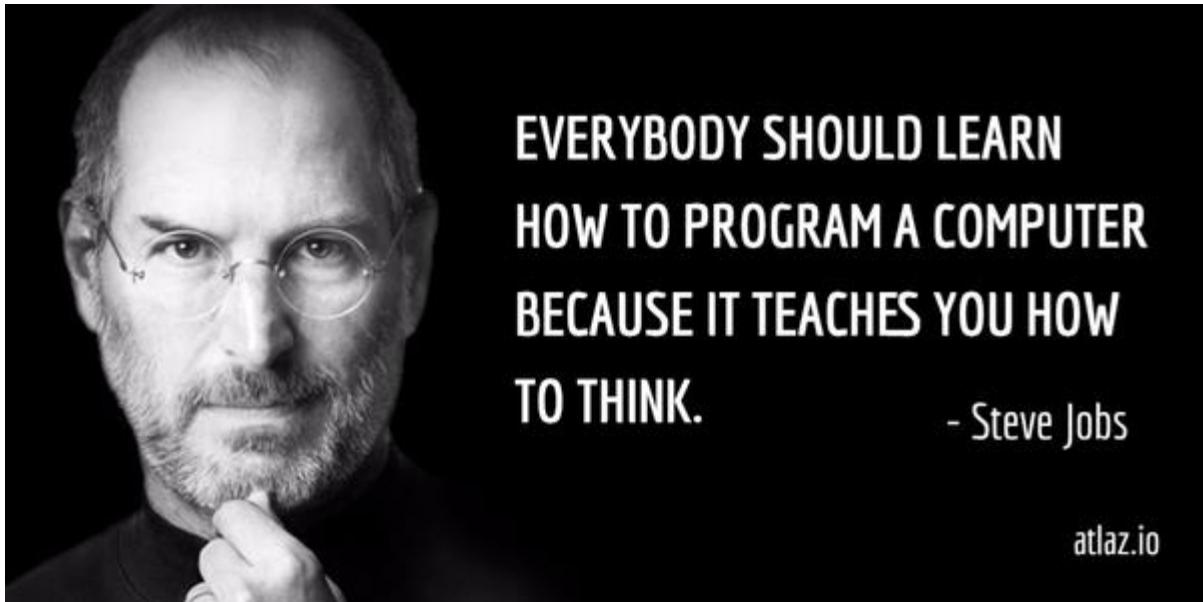
received data: b'Moein'
```

MainWindow

The screenshot shows a simple Windows application window titled "MainWindow". Inside the window, there is a text area displaying the message "Hello, World!". Below the text area is a text input field containing the text "Moein". To the right of the input field is a button labeled "Send Data". The window has standard operating system window controls (minimize, maximize, close) at the top.

# Comments





# C++

# 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
1 #include "mainwindow.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     MainWindow w;
8     w.show();
9
10    return a.exec();
11}
12

```

```

> mainwindow.cpp
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     qDebug() << "Hello world!";;
10}
11
12 MainWindow::~MainWindow()
13 {
14     delete ui;
15 }

```

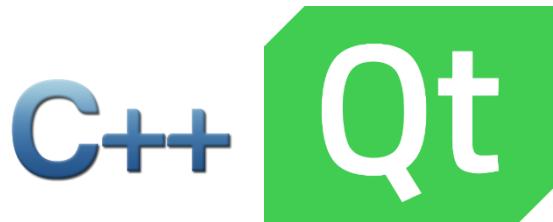
```

> mainwindow.h
1 ifndef MAINWINDOW_H
2 define MAINWINDOW_H
3
4 include <QMainWindow>
5 include <QDebug>
6
7 namespace Ui {
8 class MainWindow;
9 }
10
11 class MainWindow : public QMainWindow
12 {
13     Q_OBJECT
14
15 public:
16     explicit MainWindow(QWidget *parent = 0);
17     ~MainWindow();
18
19 private:
20     Ui::MainWindow *ui;
21 };
22
23 endif // MAINWINDOW_H

```



110 msec



3 msec

```
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

```
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 }
```

```
Application Output | ⌂ < > ▶ ■ ⌂ + -  
ArakUT  
Starting E:\Roshd\Qt\ArakUT\release\ArakUT.exe...  
3  
3682913
```

# Python for DSP

## Numpy

## 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"
```

## 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"
```



## 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]"
```



## 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}"
```

## 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"
```

## 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!"
```



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]]"
```



libraries      Numpy      SciPy      Matplotlib

Numpy      Array indexing

```
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"
```

```
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)"
```

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]])"
```

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))
```



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

```

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)

```

```

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]]"

```



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]]"
```



```
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]]
```

libraries

Numpy

SciPy

Matplotlib



Numpy

Broadcasting

```
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)
```

libraries      Numpy      SciPy      Matplotlib

## SciPy

```
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)
```

```
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)
```



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()
```



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()
```



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()
```

# Random Variables & PDF