دانشگاه صنعتی اراک

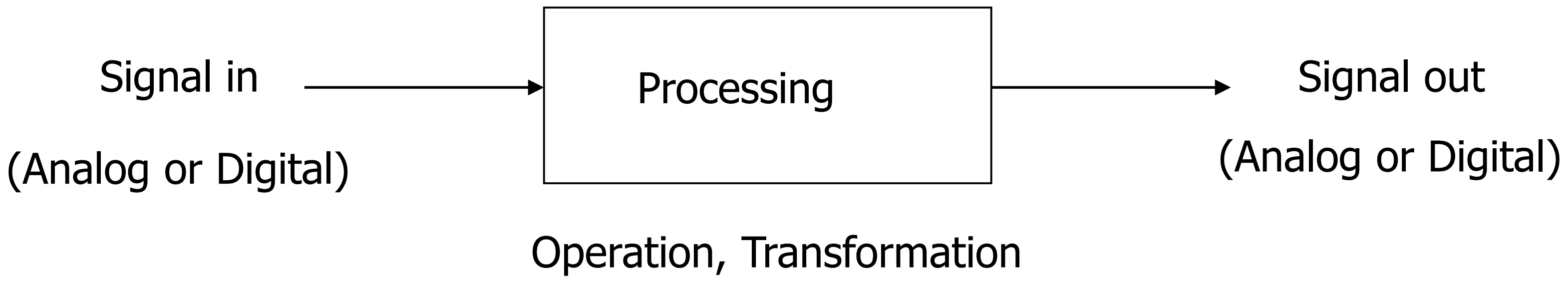
Digital Signal Processing

Arak University of Technology

By: Dr. Moein Ahmadi

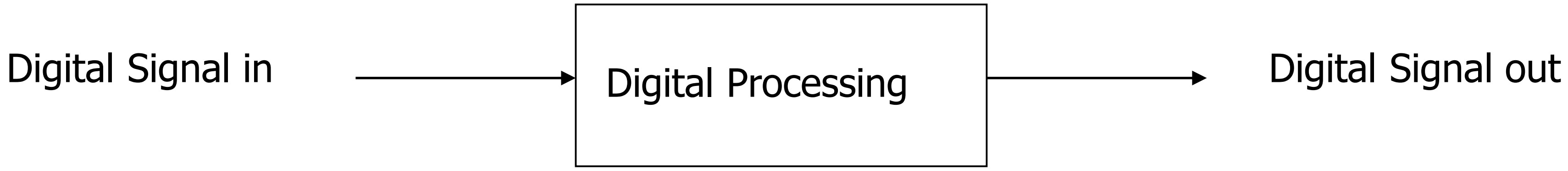
gmoein.github.io

What is Signal Processing?



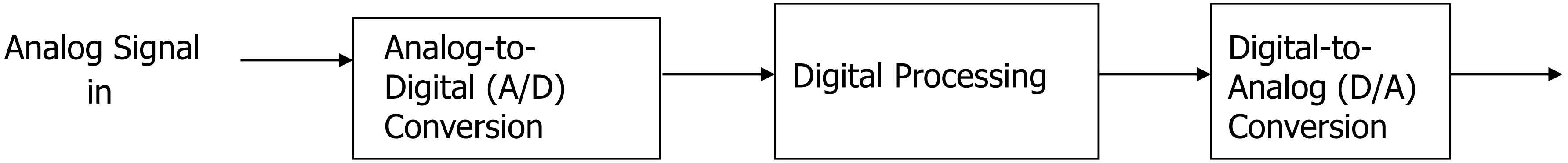
- Example of Signals:
  - Analog: Speech, Music, Photos, Video, Radar, Sonar, …
  - Discrete-domain/Digital:
    - digitized speech, digitized music, digitized images, digitized video, digitized radar and sonar signals,…
    - stock market data, daily max temperature data, ...

Digital Signal Processing and ADC

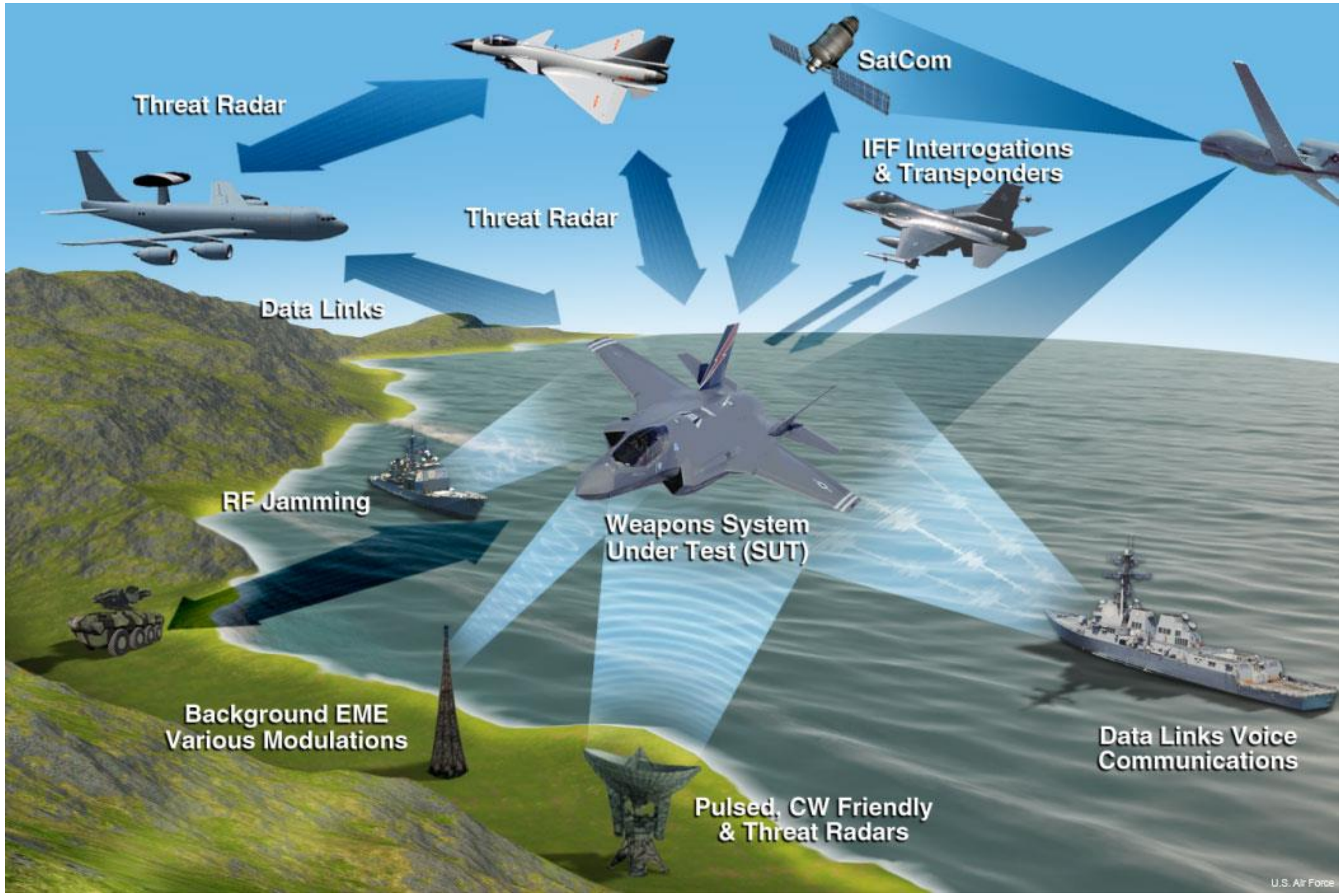


Operation, Transformation performed on digital signals (using a computer or other special-purpose digital hardware)

- But what about analog signals?


Analog Signal in
Analog-to-Digital (A/D) Conversion
Digital Processing
Digital-to-Analog (D/A) Conversion

General System- EW Environment



رادار
شنود
جمینگ
مخابرات
سنسور های مختلف

رادار: جستجو – رهگیر – تصویر برداری

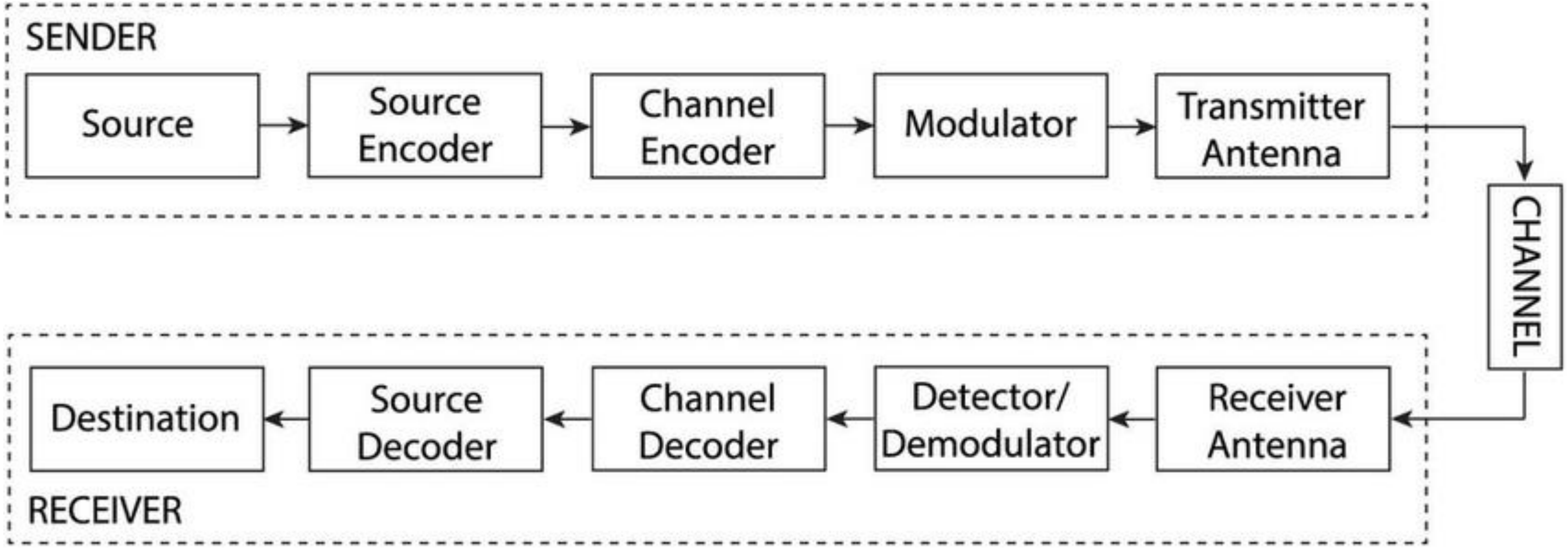
رادار جستجو:

آشکارسازی و تعیین فاصله و سرعت اهداف

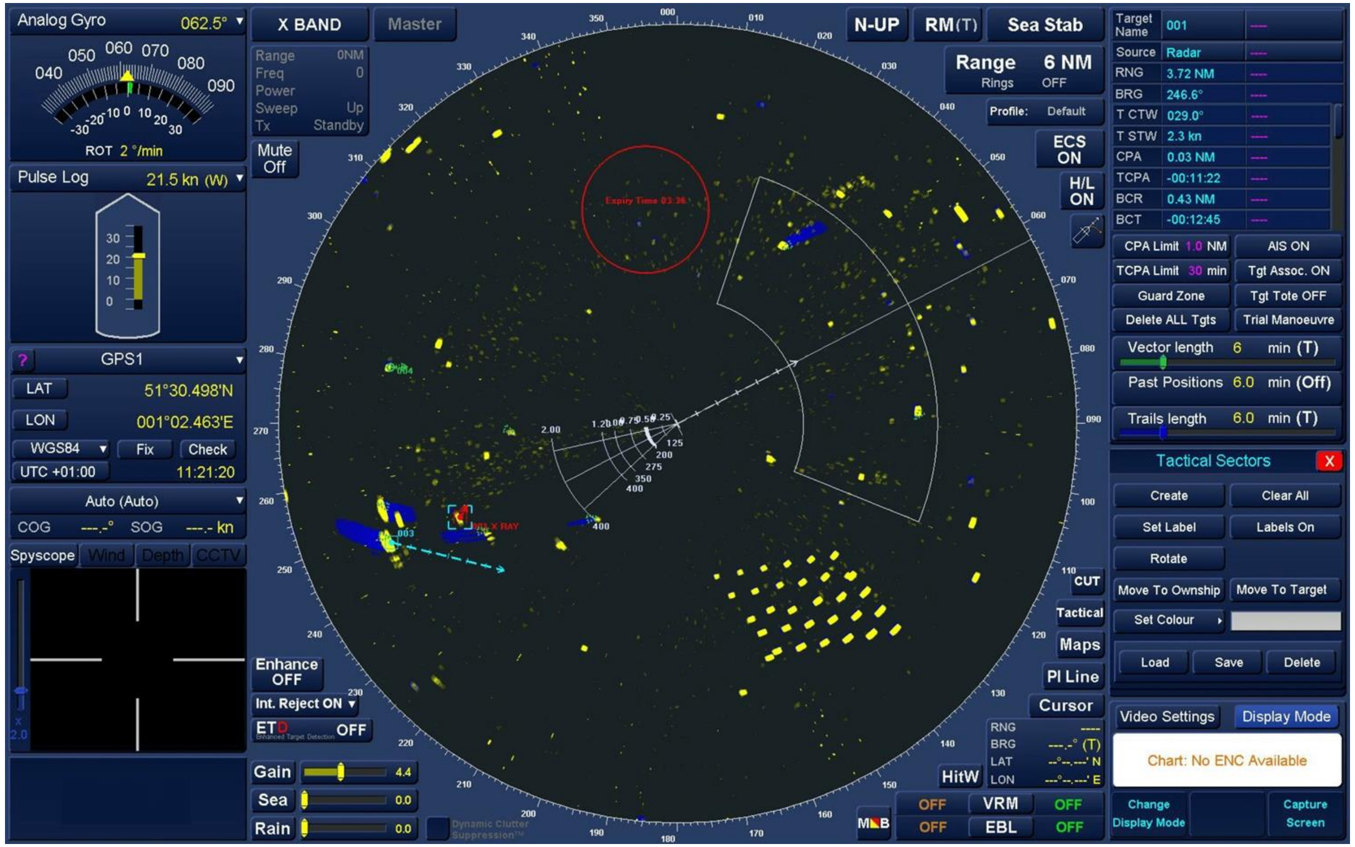
فیلتر فشرده سازی
پردازش داپلر
آشکارسازی
رهگیری حین اسکن

نمونه هایی از سیستم های مورد بحث
در پردازش سیگنال

Communications System Block Diagram



General System- Phased Array Radar



Analog Gyro 062.5°
ROT 2 °/min
Pulse Log 21.5 kn (W)
GPS1
LAT 51°30.498'N
LON 001°02.463'E
WGS84
UTC +01:00 11:21:20
Auto (Auto)
COG ---.-° SOG ---.- kn

X BAND
Range 0NM
Freq 0
Power
Sweep Up
Tx Standby

Range 6 NM
Rings OFF
Profile: Default

Target Name	001	----
Source	Radar	----
RNG	3.72 NM	----
BRG	246.6°	----
T CTW	029.0°	----
T STW	2.3 kn	----
CPA	0.03 NM	----
TCPA	-00:11:22	----
BCR	0.43 NM	----
BCT	-00:12:45	----

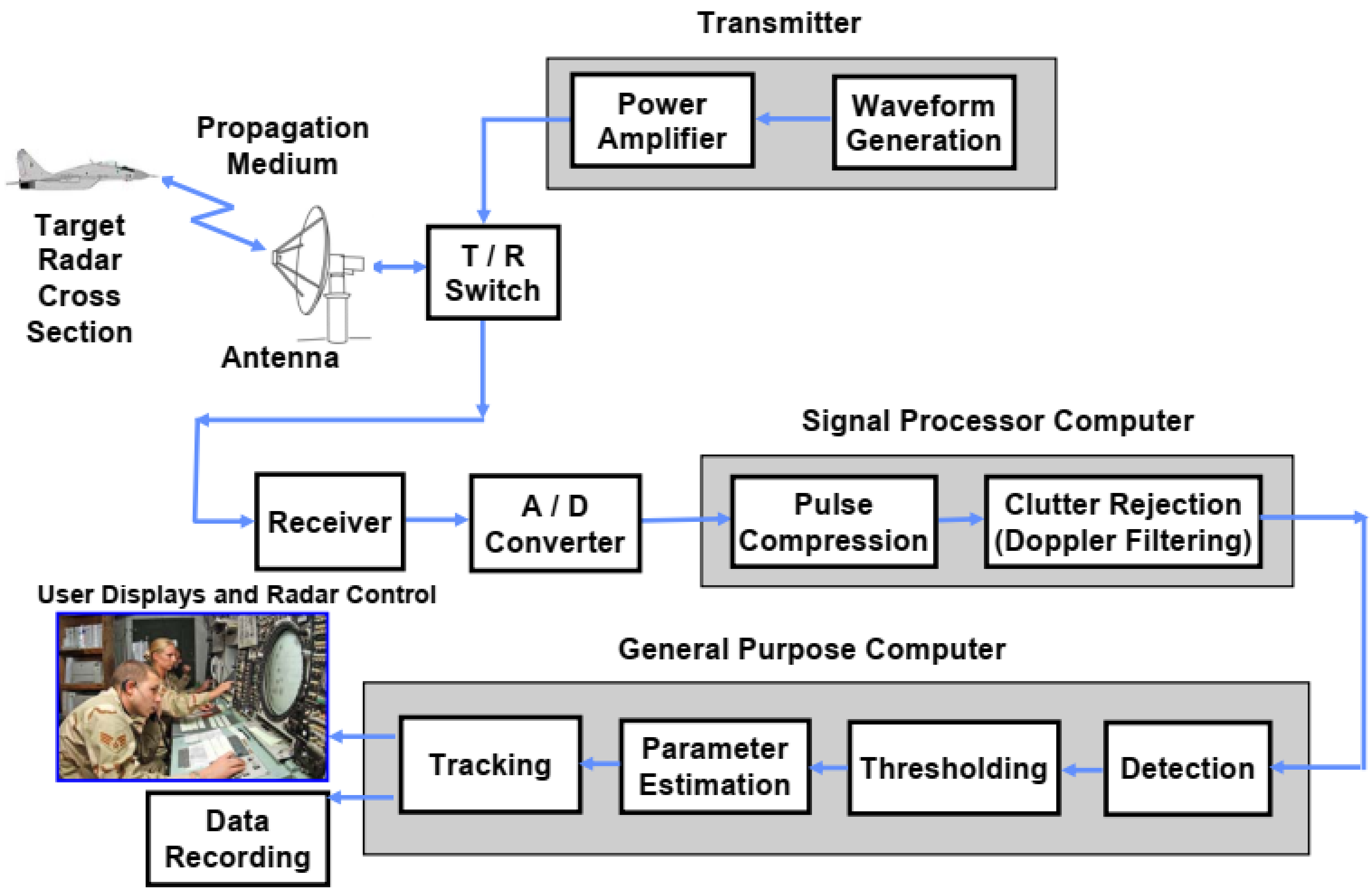
CPA Limit 1.0 NM
TCPA Limit 30 min
Vector length 6 min (T)
Past Positions 6.0 min (Off)
Trails length 6.0 min (T)

RNG ----
BRG ---.-° (T)
LAT --°--.---' N
LON ---°--.---' E

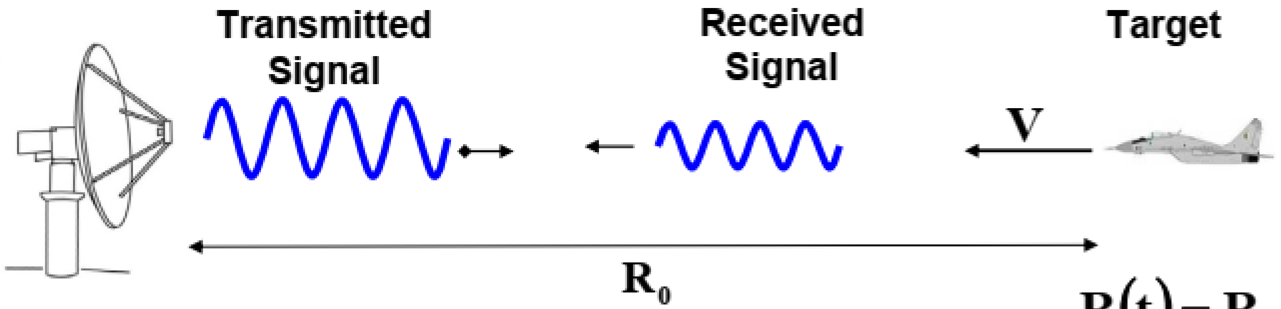
Gain 4.4
Sea 0.0
Rain 0.0

Chart: No ENC Available

Example- Radar System



Example- Radar Basics



$$R(t) = R_0 - Vt$$

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

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

Amplitude

Depends on RCS, radar parameters, range, etc.

Angle

Azimuth and Elevation

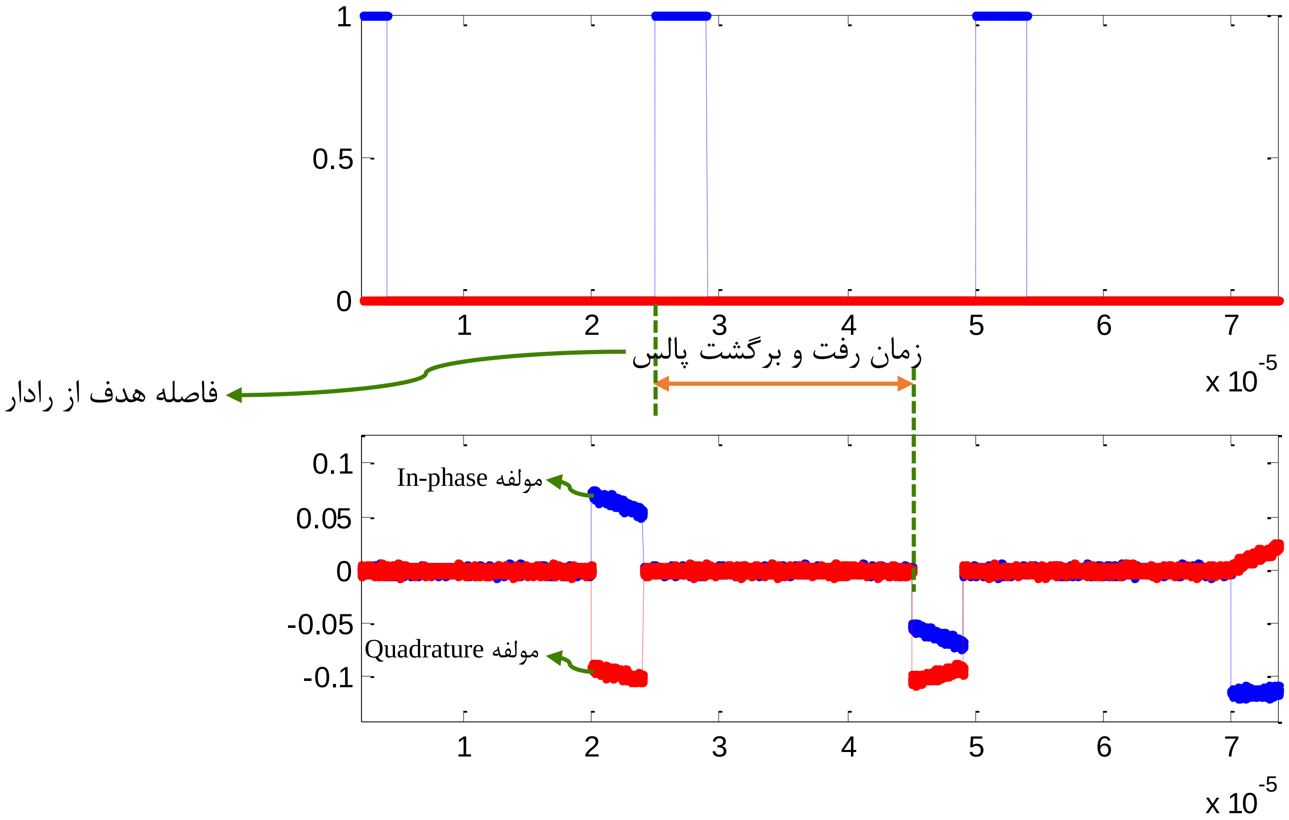
Time Delay

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

Doppler Frequency

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

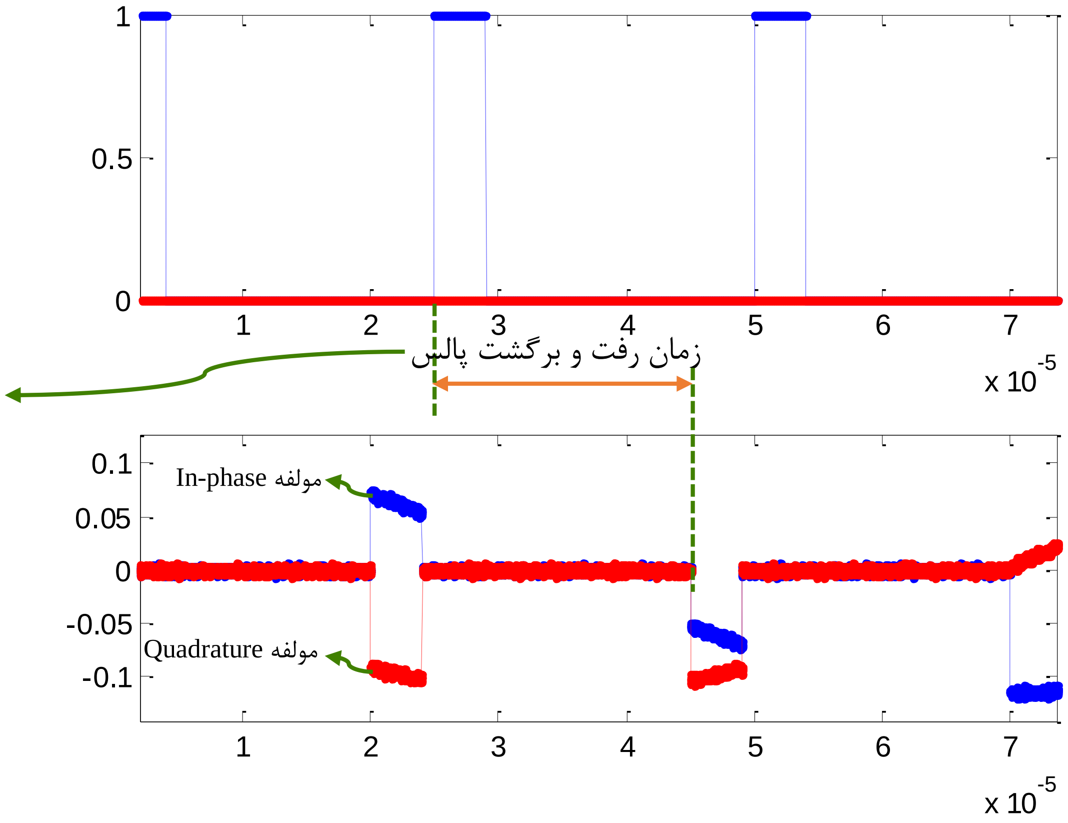
Radar Basics



سیگنال ارسالی
از فرستنده رادار

سیگنال دریافتی از
هدف در گیرنده رادار

Radar Basics

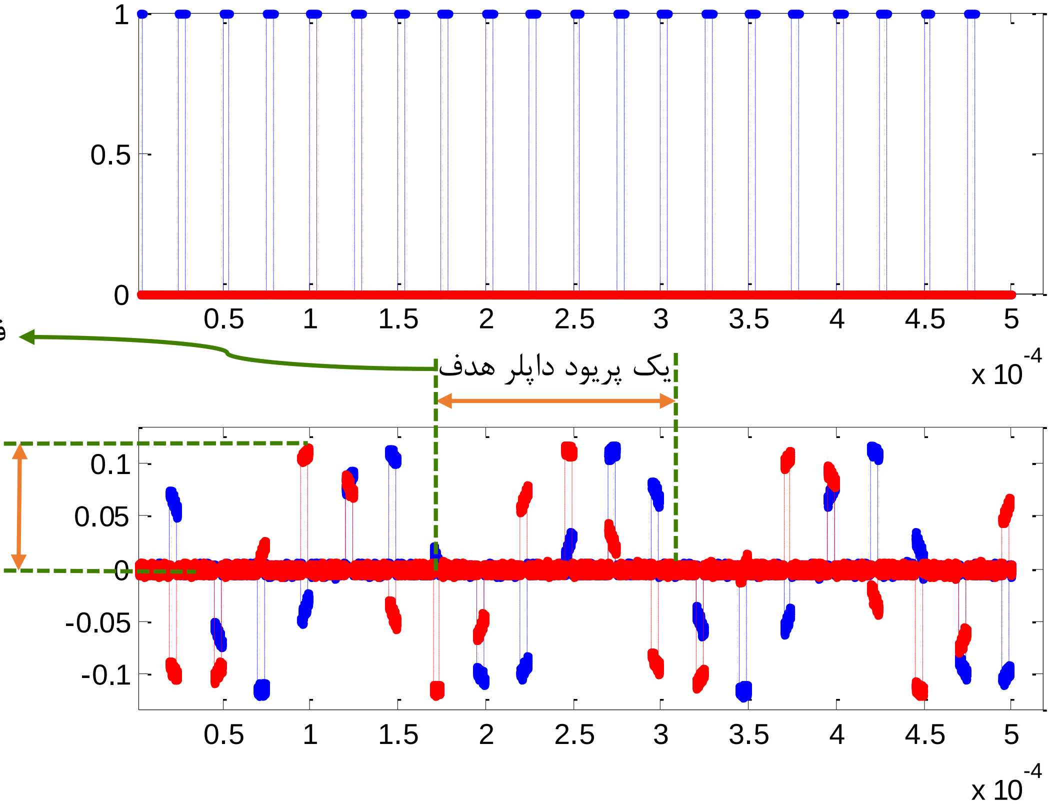


سیگنال ارسالی
از فرستنده رادار

فاصله هدف از رادار

سیگنال دریافتی از
هدف در گیرنده رادار

Radar Basics



سیگنال ارسالی
از فرستنده رادار

فرکانس داپلر (سرعت هدف)

یک پریود داپلر هدف

دامنه سیگنال دریافتی از هدف

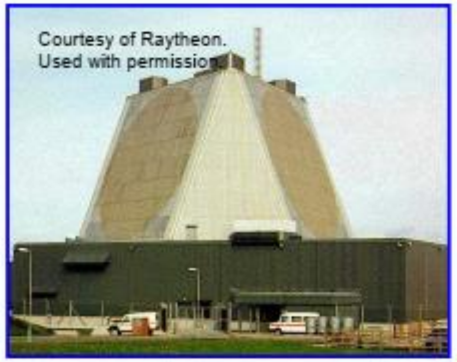
سیگنال دریافتی از
هدف در گیرنده رادار

Surveillance and Fire Control Radars

Courtesy of US Air Force
Used with permission.

Courtesy of NATO.

Photo courtesy of ITT Corporation. Used with permission.



Courtesy of Raytheon.
Used with permission.

Courtesy of US Navy.

Courtesy of Raytheon. Used with permission.

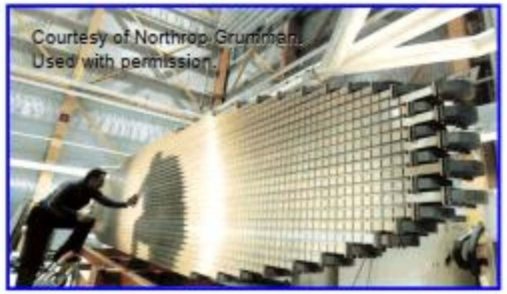
Airborne Radars

Courtesy of US Air Force.

Courtesy of US Air Force

Courtesy of Boeing
Used with permission

Courtesy of Northrop Grumman.
Used with permission.



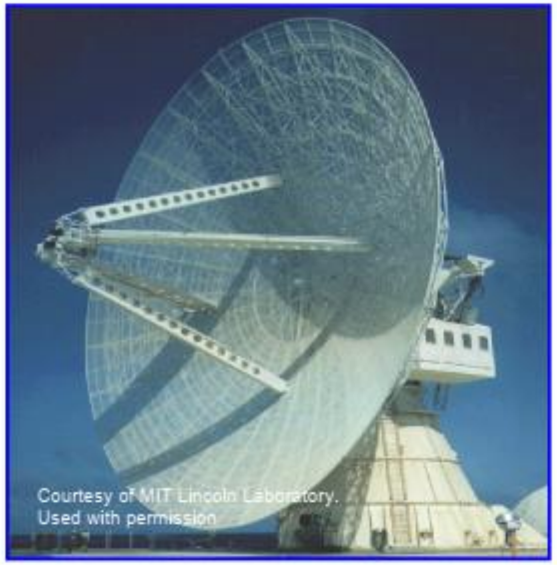
Courtesy of Northrop Grumman.
Used with permission.

Courtesy of US Air Force.

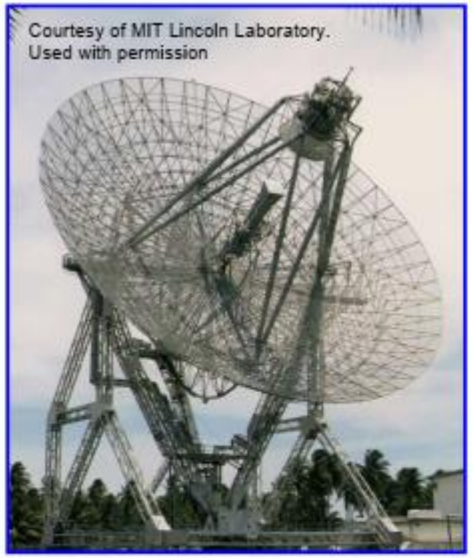
Courtesy of US Navy.

Courtesy of Raytheon
Used with permission

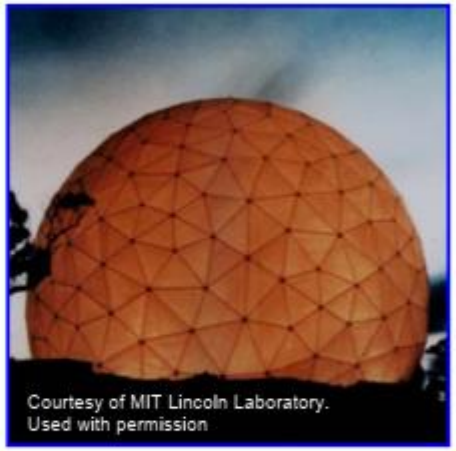
Instrumentation Radars



Courtesy of MIT Lincoln Laboratory.
Used with permission



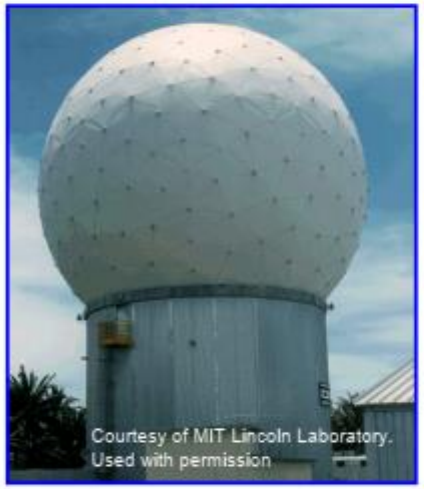
Courtesy of MIT Lincoln Laboratory.
Used with permission



Courtesy of MIT Lincoln Laboratory.
Used with permission

Courtesy of Lockheed Martin

Courtesy of MIT Lincoln Laboratory.
Used with permission

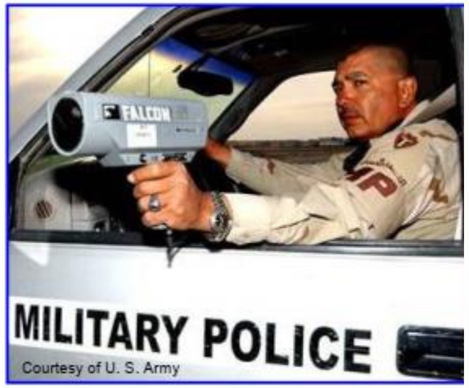


Courtesy of MIT Lincoln Laboratory.
Used with permission

Civil Radars

Courtesy of Target Corporation

Courtesy of Northrop Grumman. Used with permission.



Courtesy of U. S. Army

Courtesy of NASA

Courtesy of NASA

Courtesy of akseabird..

Courtesy of NOAA

Courtesy of FAA

Courtesy of MIT Lincoln Laboratory Used with permission

Courtesy of NASA

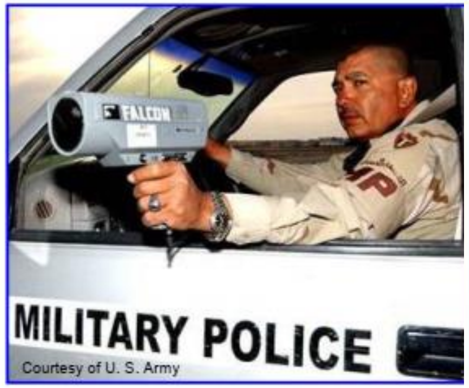
Courtesy of Raymarine. Used with permission.

Courtesy of NASA

Civil Radars

Courtesy of Target Corporation

Courtesy of Northrop Grumman. Used with permission.



Courtesy of U. S. Army

Courtesy of NASA

Courtesy of NASA

Courtesy of akseabird..

Courtesy of NOAA

Courtesy of FAA

Courtesy of MIT Lincoln Laboratory
Used with permission

Courtesy of NASA

Courtesy of Raymarine.
Used with permission.

Courtesy of NASA

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

ICASSP 2019

Signal Processing Topics

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

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

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

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

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

Signal Processing Topics

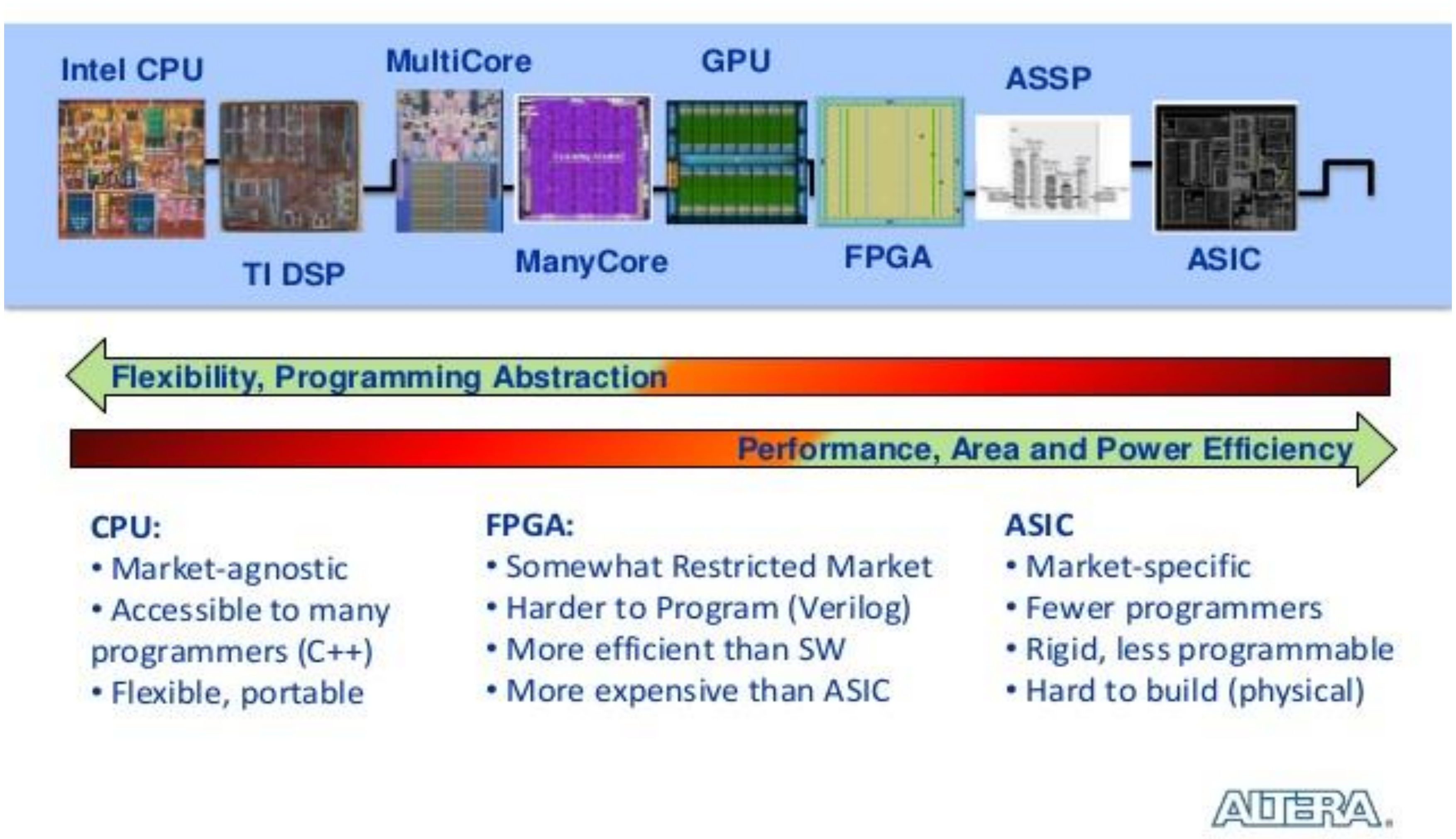
- Adaptive beamforming
- Array processing for biomedical applications
- Array processing for communications
- Big data
- Blind source separation and channel identification
- Computational and optimization techniques
- Compressive sensing and sparsity-based signal processing
- Detection and estimation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

IEEE SAM 2018 SHEFFIELD (UK)

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


Intel CPU
TI DSP
MultiCore
ManyCore
GPU
FPGA
ASSP
ASIC
Flexibility, Programming Abstraction
Performance, Area and Power Efficiency
ALTERA.

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)

Programming Languages for Signal Processing Course (Algorithm Design)

MATLAB

Anaconda Pip conda

Python 2.7 and 3.5.

```
python --version
```

Basic data types:

Numbers, Booleans, Strings

Containers:

Lists [,,]

Dictionaries {“:”,”:”}

Sets {'', “}

Tuples (“,”) **Slicing, Loops**

Functions

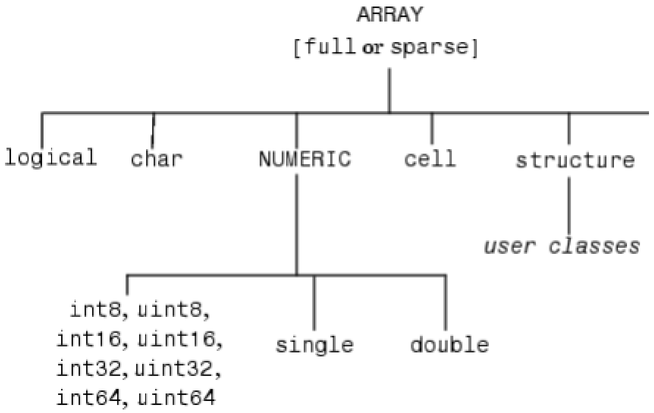
Classes

Libraries Numpy SciPy

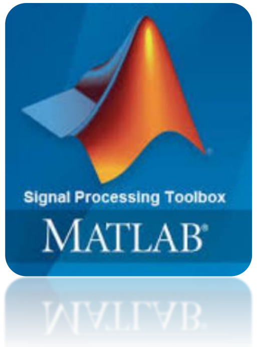
Matplotlib

https://www.mathworks.com

Basic data types:

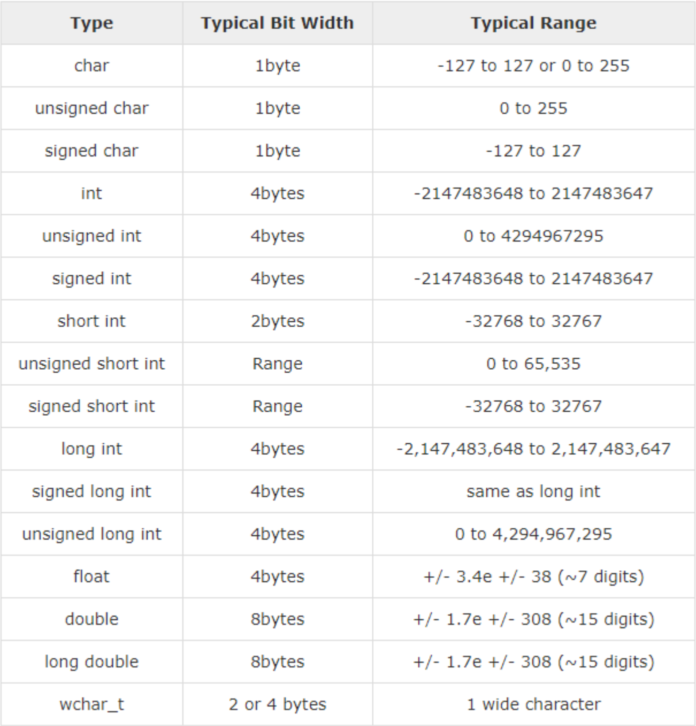


Toolbox



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

Basic data types:



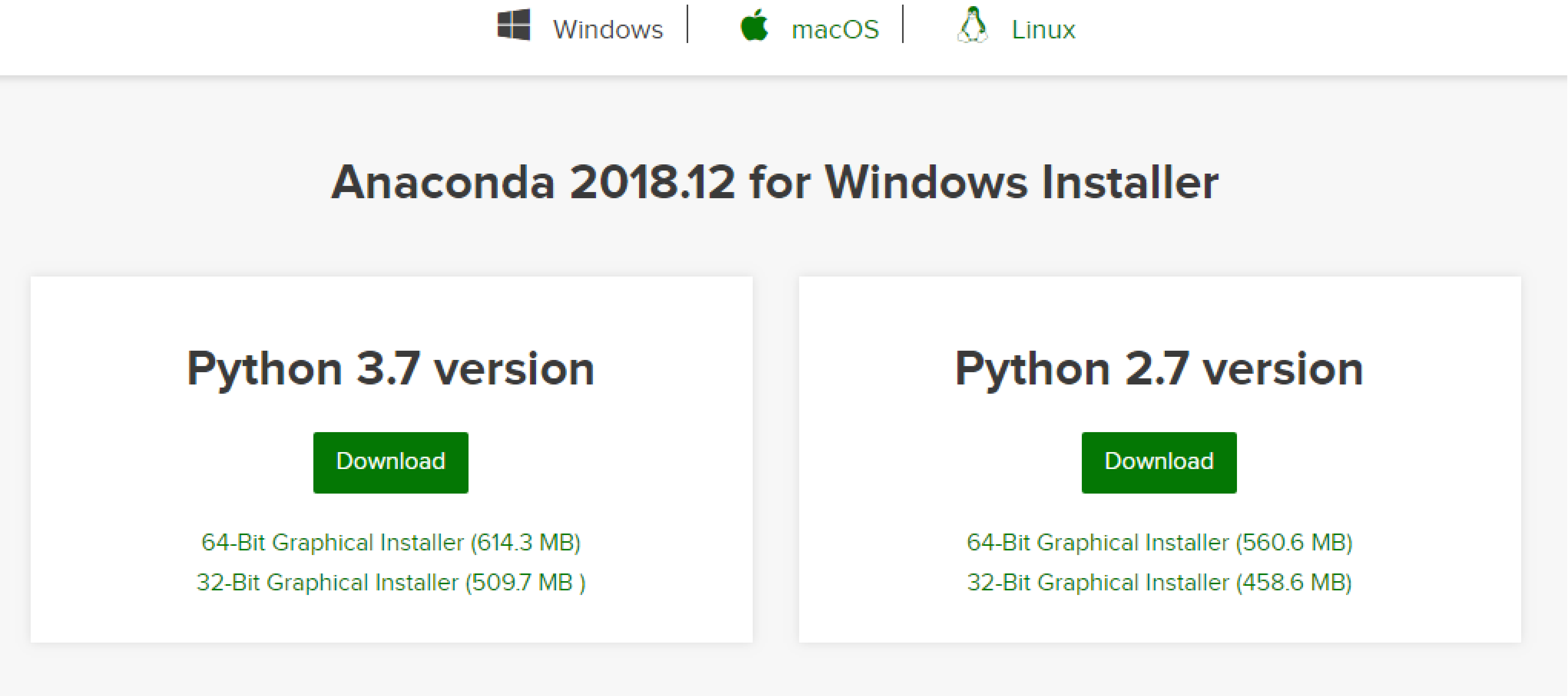
Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range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

Containers: QVector , …

Classes

Libraries

Getting Started With Python Using Anaconda



Anaconda 2018.12 for Windows Installer

Python 3.7 version

64-Bit Graphical Installer (614.3 MB)

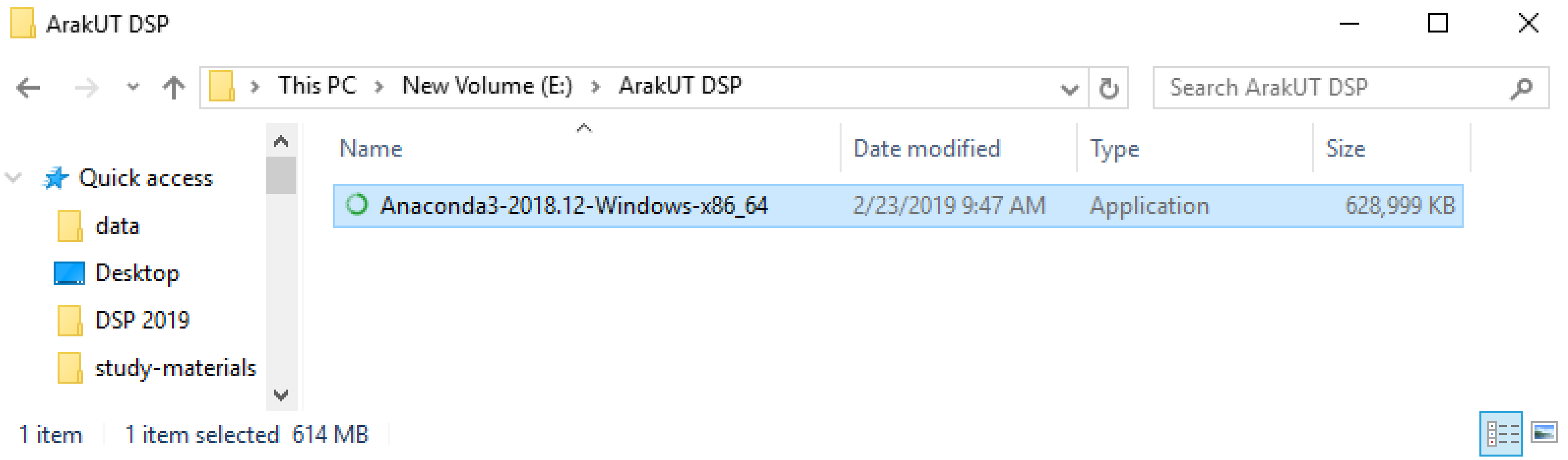
32-Bit Graphical Installer (509.7 MB)

Python 2.7 version

64-Bit Graphical Installer (560.6 MB)

32-Bit Graphical Installer (458.6 MB)

Getting Started With Python Using Anaconda



ArakUT DSP

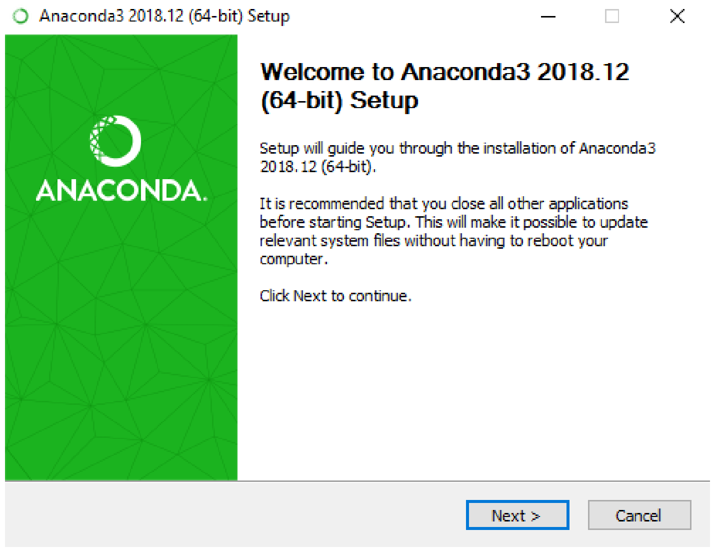
This PC › New Volume (E:) › ArakUT DSP

Search ArakUT DSP

Quick access
- data
- Desktop
- DSP 2019
- study-materials

Name	Date modified	Type	Size
Anaconda3-2018.12-Windows-x86_64	2/23/2019 9:47 AM	Application	628,999 KB

1 item 1 item selected 614 MB



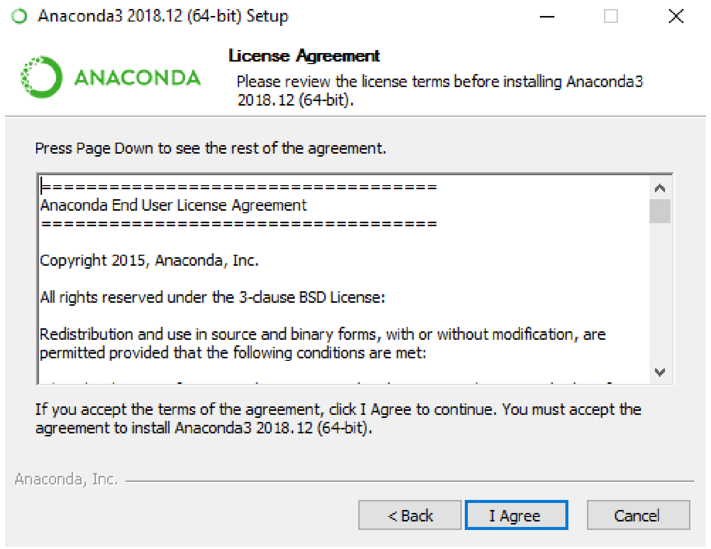
Anaconda3 2018.12 (64-bit) Setup

Welcome to Anaconda3 2018.12 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2018.12 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.



Anaconda3 2018.12 (64-bit) Setup

License Agreement

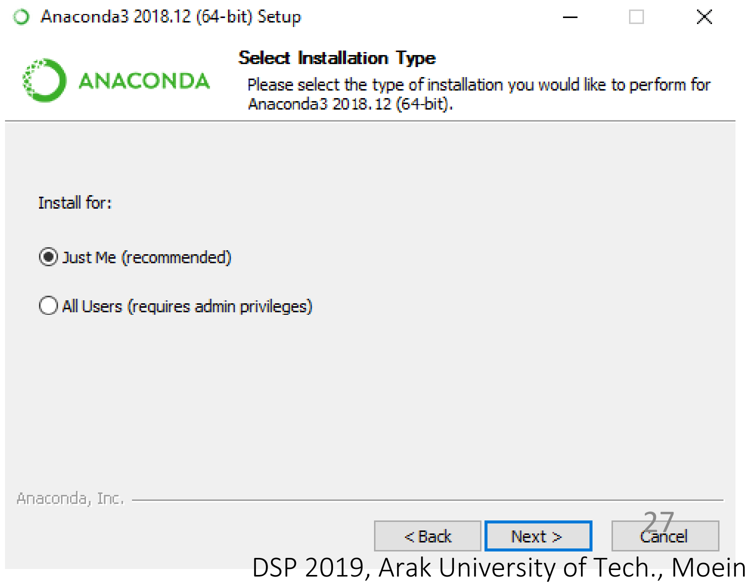
Please review the license terms before installing Anaconda3 2018.12 (64-bit).

Press Page Down to see the rest of the agreement.

```
===================================
Anaconda End User License Agreement
===================================

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2018.12 (64-bit).

Anaconda, Inc.



Anaconda3 2018.12 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2018.12 (64-bit).

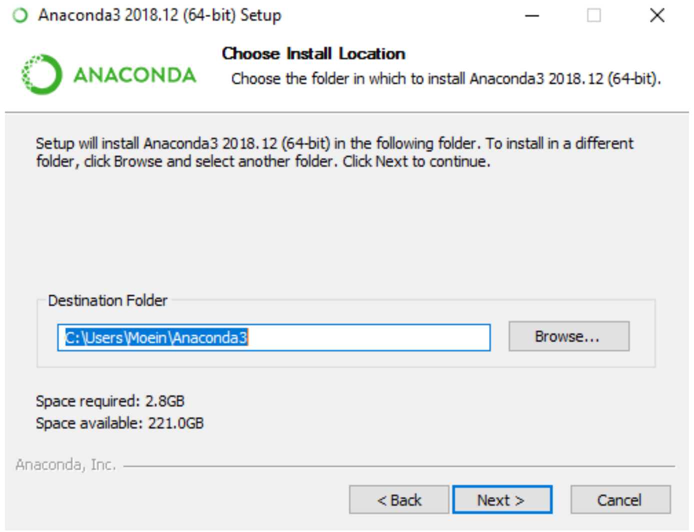
Install for:

● Just Me (recommended)

○ All Users (requires admin privileges)

Anaconda, Inc.

Getting Started With Python Using Anaconda



Choose Install Location

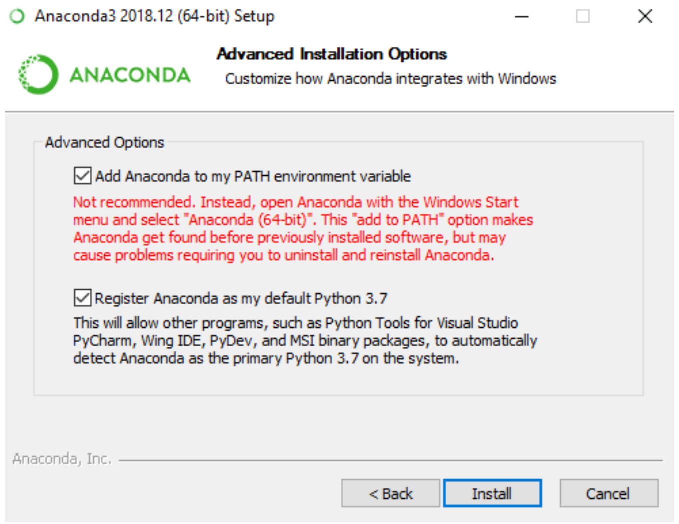
Choose the folder in which to install Anaconda3 2018.12 (64-bit).

Setup will install Anaconda3 2018.12 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Users\Moein\Anaconda3

Space required: 2.8GB
Space available: 221.0GB

Anaconda, Inc.



Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

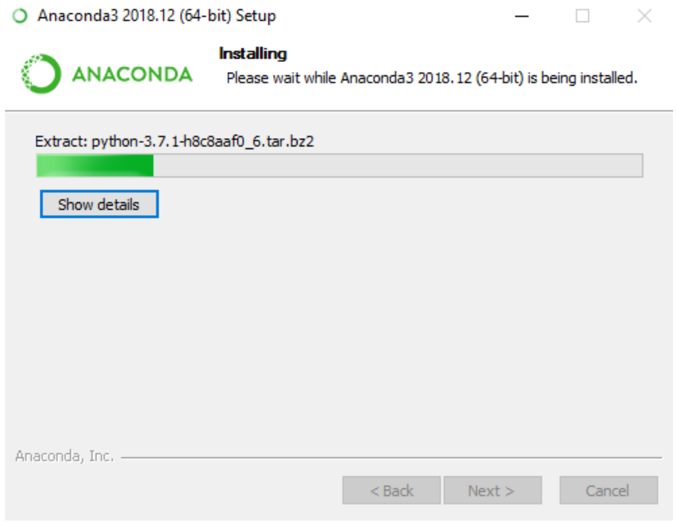
☑ Add Anaconda to my PATH environment variable

Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda as my default Python 3.7

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.

Anaconda, Inc.

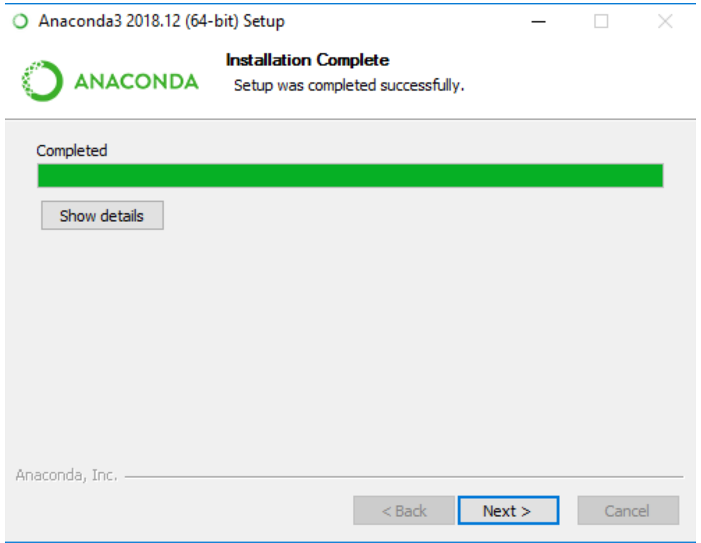


Installing

Please wait while Anaconda3 2018.12 (64-bit) is being installed.

Extract: python-3.7.1-h8c8aaf0_6.tar.bz2

Anaconda, Inc.

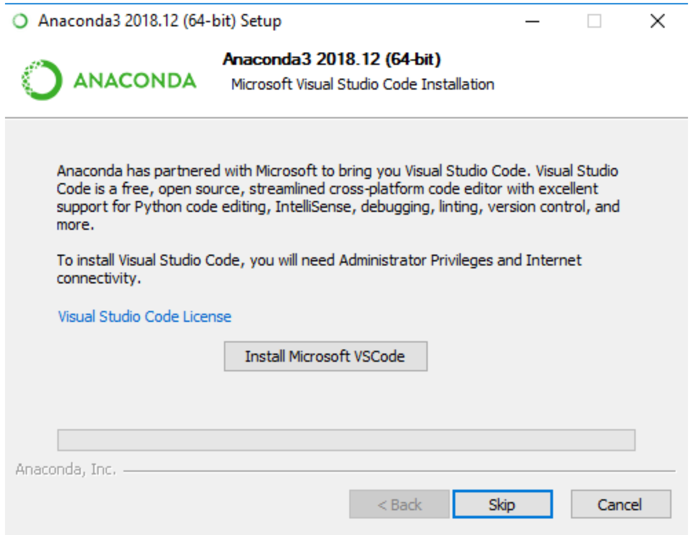


Installation Complete

Setup was completed successfully.

Completed

Anaconda, Inc.



Anaconda3 2018.12 (64-bit)

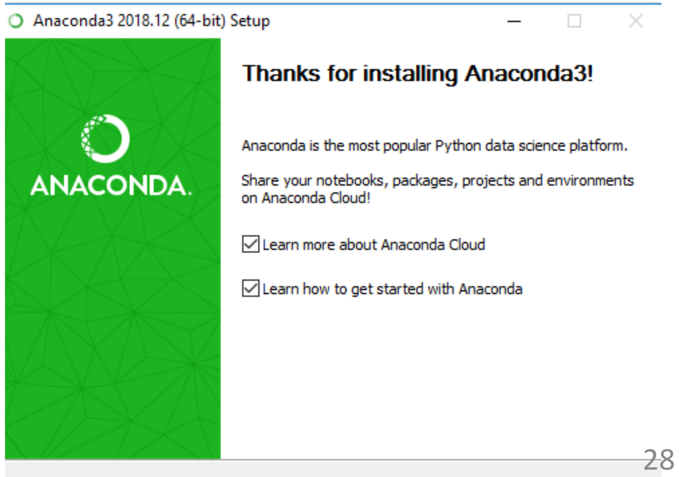
Microsoft Visual Studio Code Installation

Anaconda has partnered with Microsoft to bring you Visual Studio Code. Visual Studio Code is a free, open source, streamlined cross-platform code editor with excellent support for Python code editing, IntelliSense, debugging, linting, version control, and more.

To install Visual Studio Code, you will need Administrator Privileges and Internet connectivity.

Visual Studio Code License

Anaconda, Inc.



Thanks for installing Anaconda3!

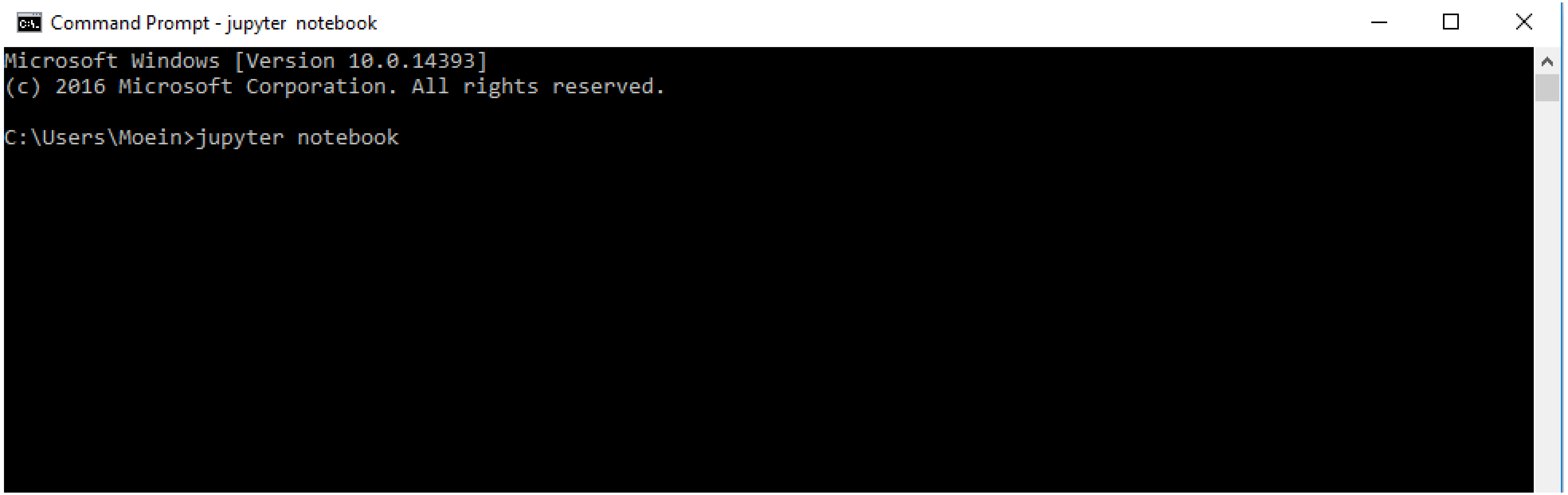
Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

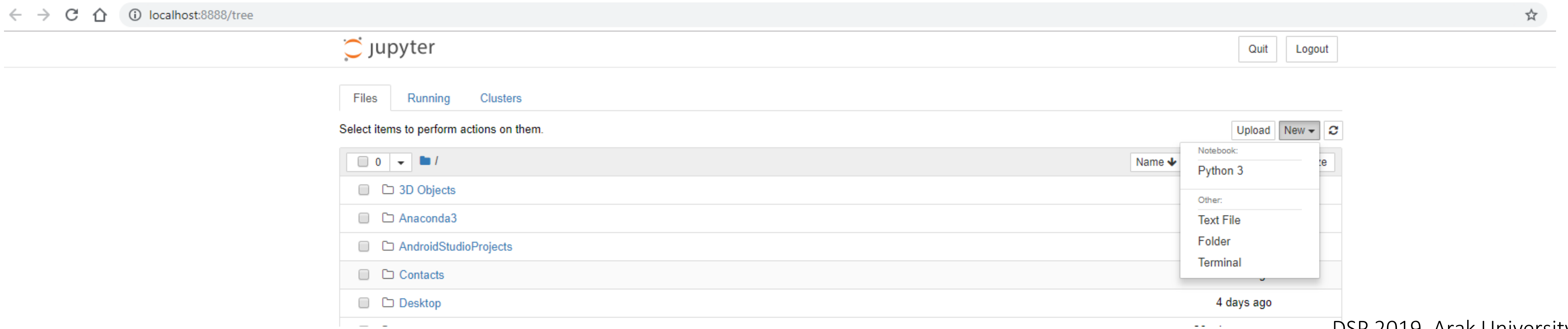
☑ Learn more about Anaconda Cloud

☑ Learn how to get started with Anaconda

Getting Started With Python Using Anaconda

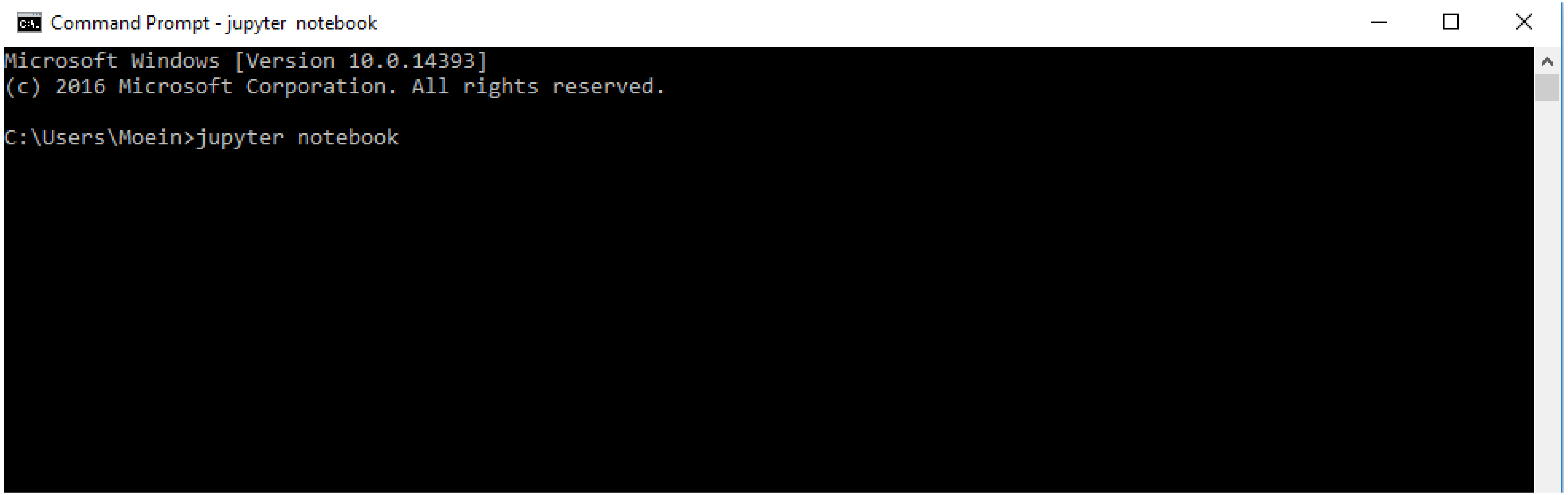


```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Moein>jupyter notebook
```



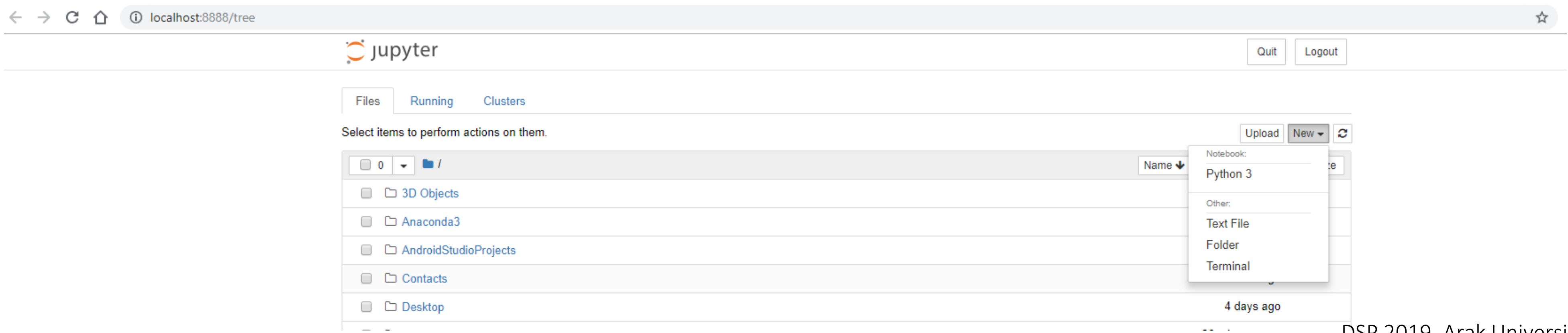
Select items to perform actions on them.

Getting Started With Python Using Anaconda



Command Prompt - jupyter notebook

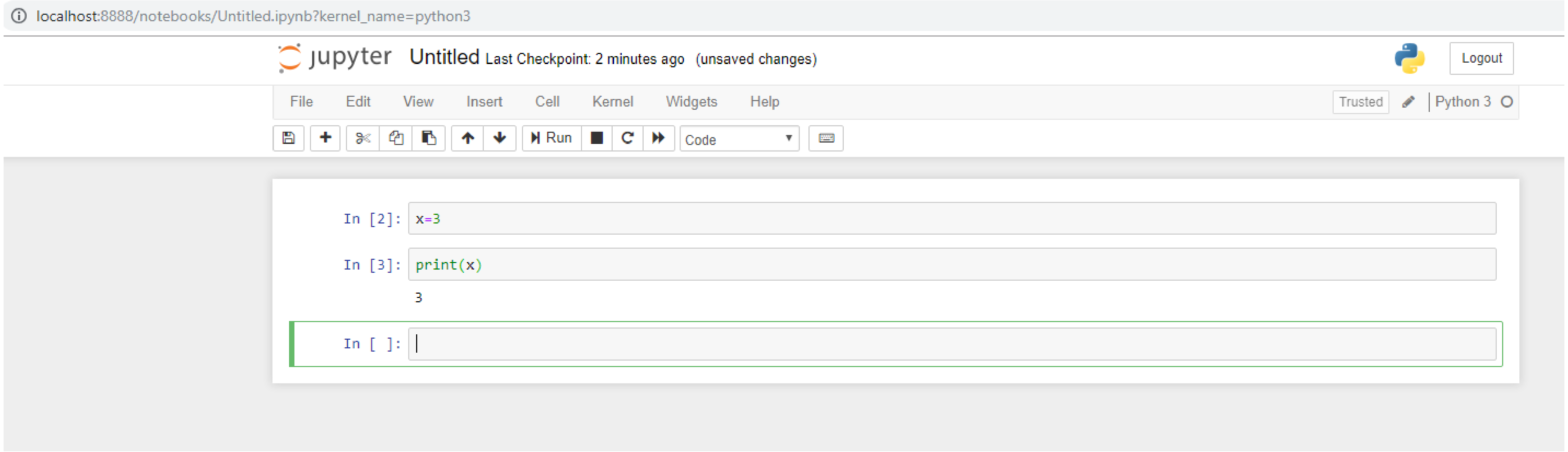
```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Moein>jupyter notebook
```



Select items to perform actions on them.

- 3D Objects
- Anaconda3
- AndroidStudioProjects
- Contacts
- Desktop 4 days ago

Getting Started With Python Using Anaconda

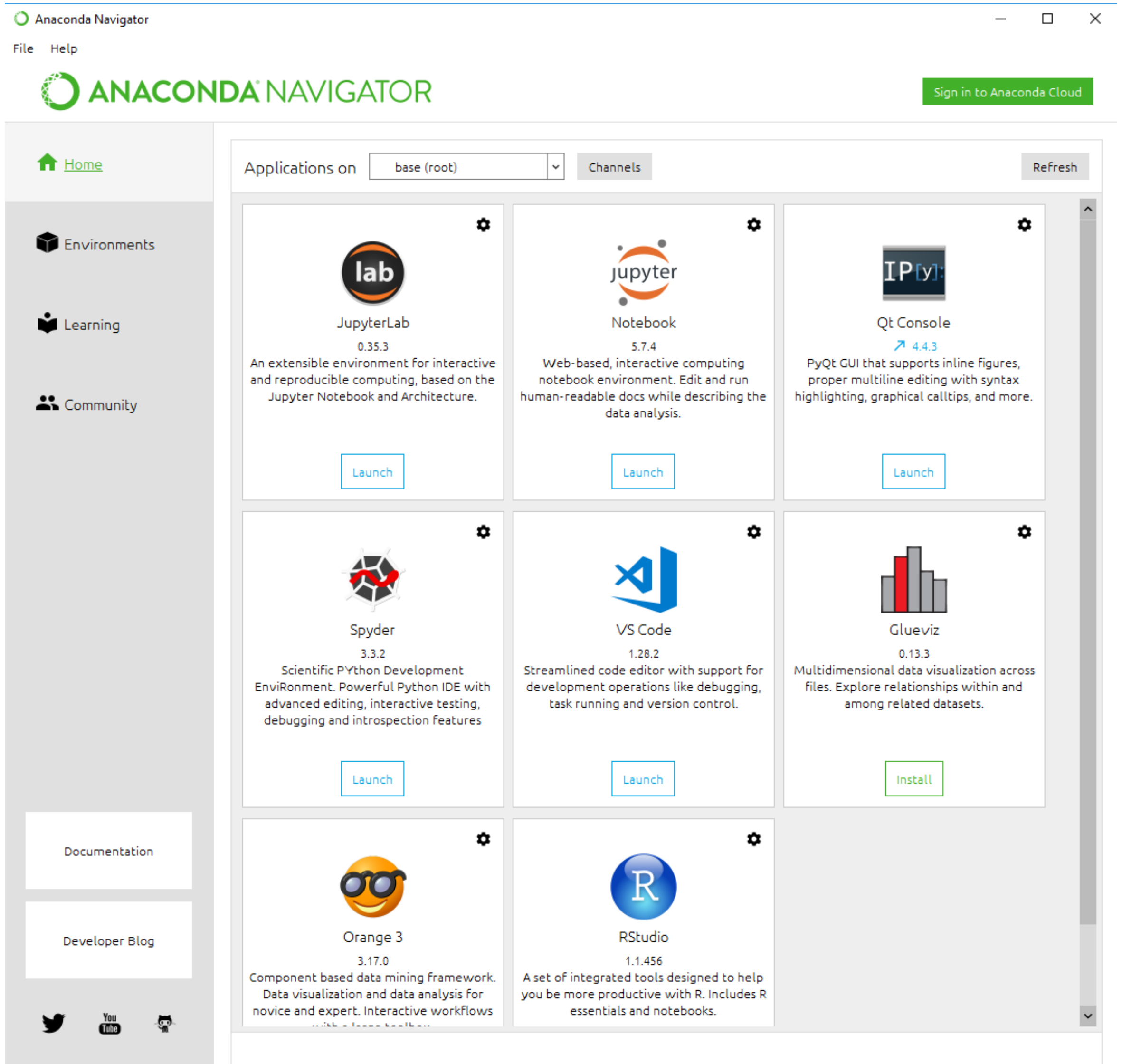


localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3
jupyter Untitled Last Checkpoint: 2 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

```
In [2]: x=3

In [3]: print(x)

        3

In [ ]: 
```

Run: Shift+Enter

Getting Started With Python Using Anaconda



Applications on base (root)

JupyterLab 0.35.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook 5.7.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console ↗ 4.4.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

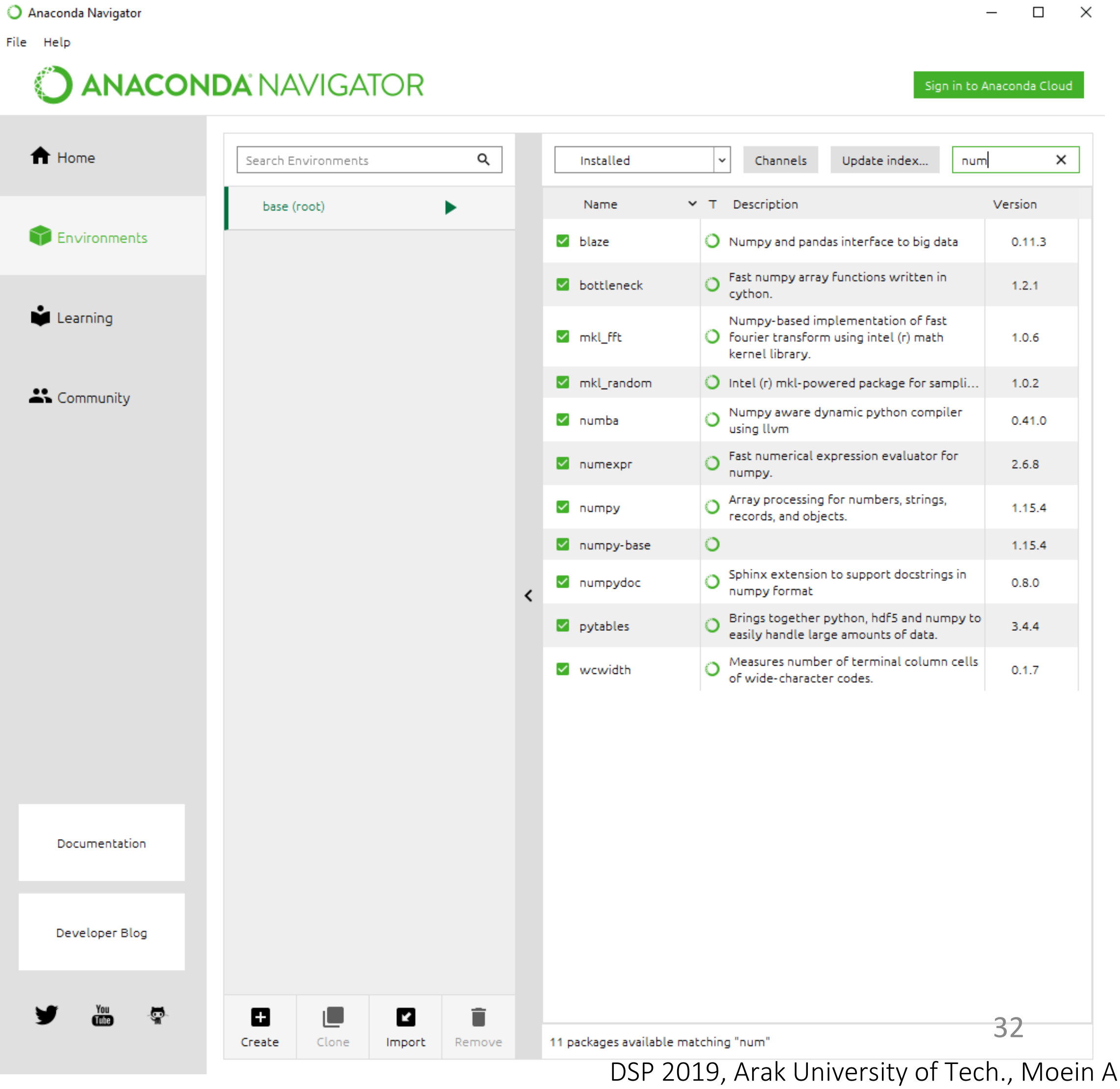
Spyder 3.3.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code 1.28.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz 0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3 3.17.0
Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows

RStudio 1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.



Name	T	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 sampli...	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

11 packages available matching "num"

Welcome to Python!

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

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

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

First Program

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

Simple Operations

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

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

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

Floats

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

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

Exponentiation

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

Quotient & Remainder

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

Strings

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

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

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

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

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

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

Input

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

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

Type Conversion

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

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

Variables

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

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

```
>>> this_is_a_normal_name = 7

>>> 123abc = 7
SyntaxError: invalid syntax

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

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

Variables

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

In-Place Operators

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

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

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

Comparisons: Booleans

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

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


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

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


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

if Statements

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

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

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

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

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

Boolean Logic

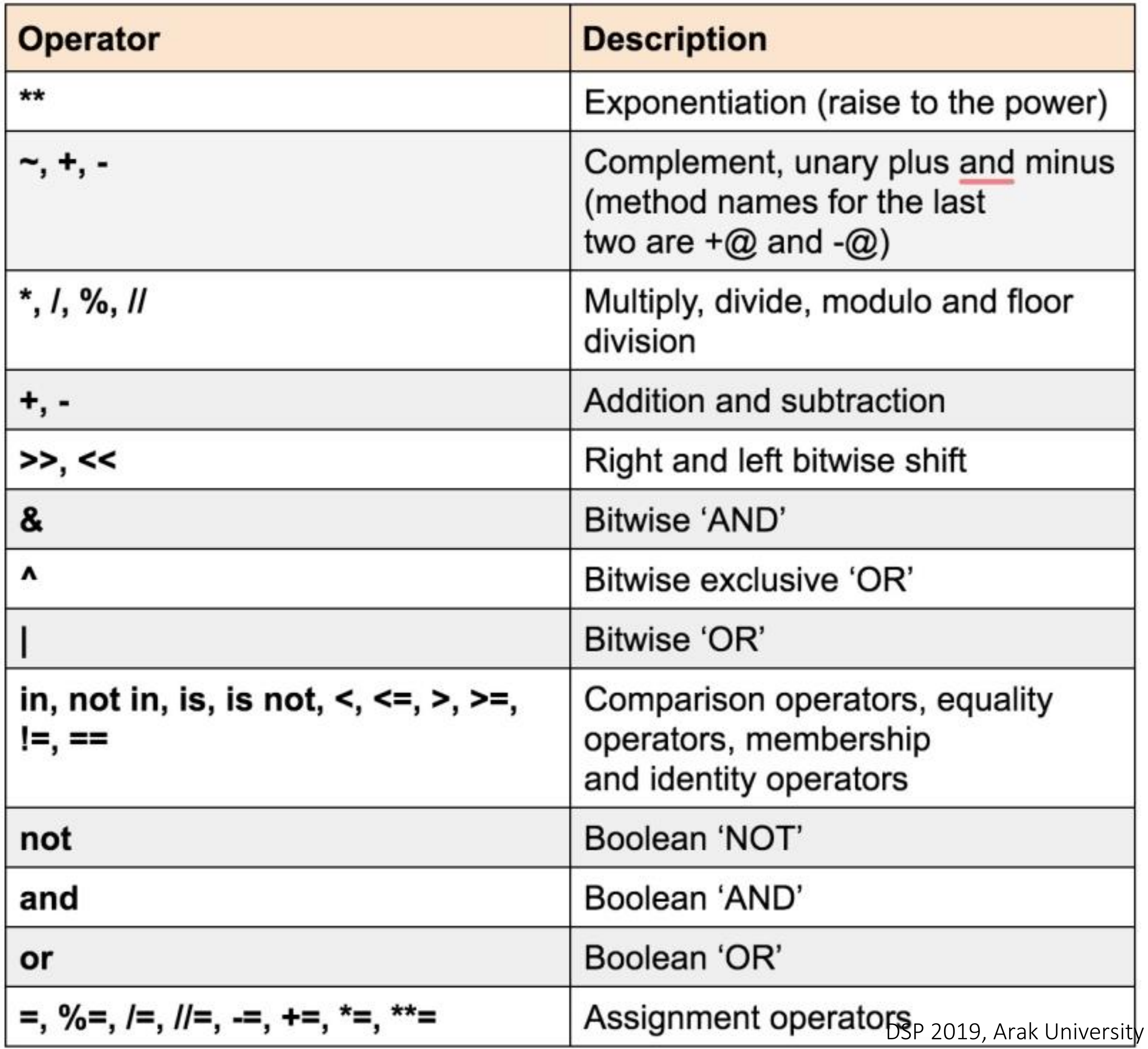
```
>>> 1 == 1 and 2 == 2
True
>>> 1 == 1 and 2 == 3
False
>>> 1 != 1 and 2 == 2
False
>>> 2 < 1 and 3 > 6
False
```

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

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

Operator Precedence

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



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

while Loops

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

print("Finished!")
```

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

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

print("Finished")
```

Lists

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

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

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

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

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

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

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

Lists

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

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

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

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

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

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

Range

```
numbers = list(range(10))
print(numbers)                               [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
numbers = list(range(3, 8))
print(numbers)                                  [3, 4, 5, 6, 7]

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

```
numbers = list(range(5, 20, 2))              [5, 7, 9, 11, 13, 15, 17, 19]
print(numbers)
```

python™

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()
```

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

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

None

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

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

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

```
Hi!
None
```

Dictionaries

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

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

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

Dictionaries

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

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

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

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

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

Tuples

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

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

List Slices

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

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

List Slices

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

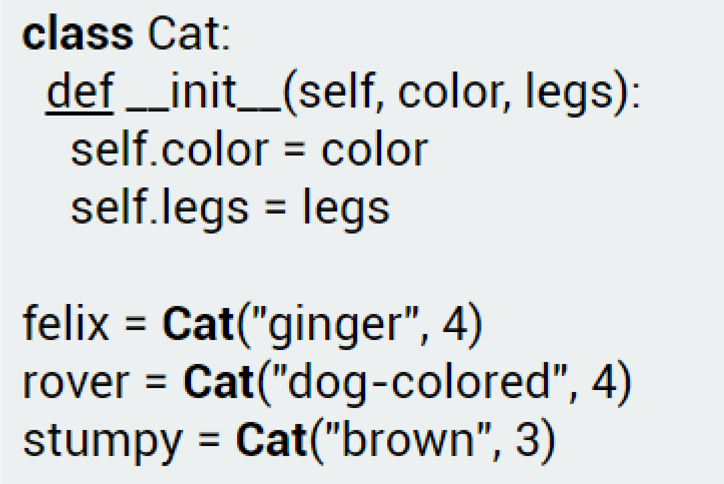
python™

String Functions

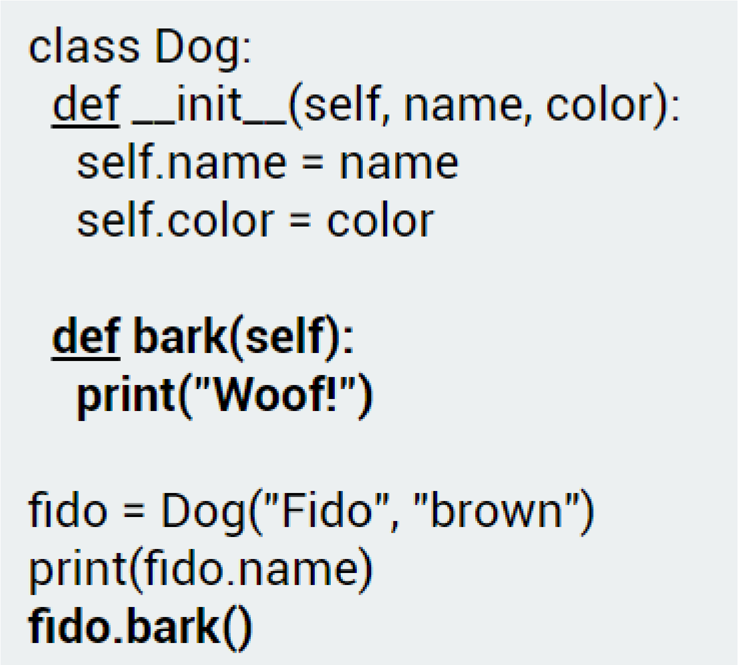
```
print(", ".join(["spam", "eggs", "ham"]))
#prints "spam, eggs, ham"

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



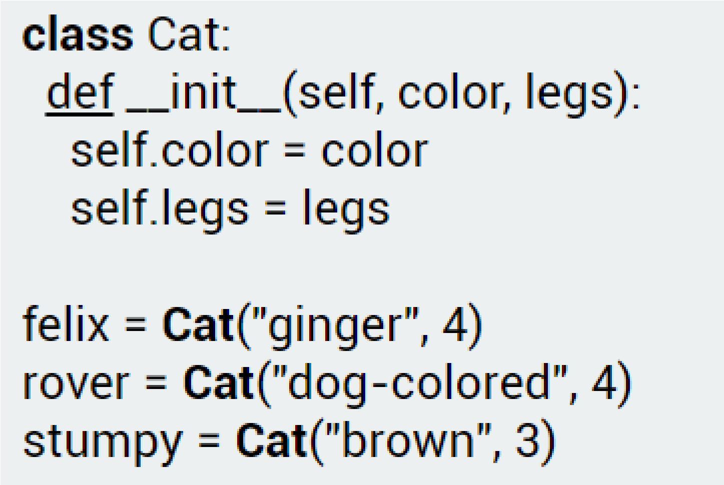
```
class Cat:
  def __init__(self, color, legs):
    self.color = color
    self.legs = legs

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



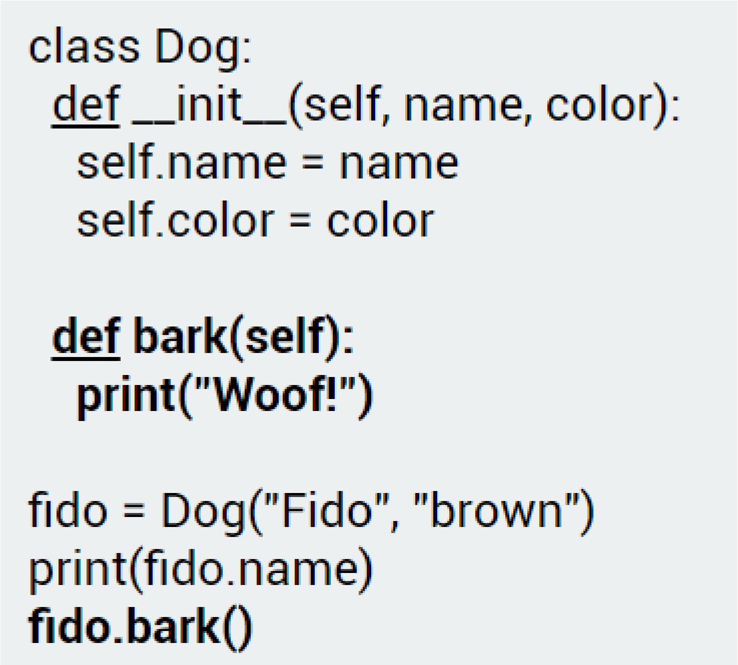
```
class Dog:
  def __init__(self, name, color):
    self.name = name
    self.color = color

  def bark(self):
    print("Woof!")

fido = Dog("Fido", "brown")
print(fido.name)
fido.bark()
```

object-oriented programming (OOP)

Classes



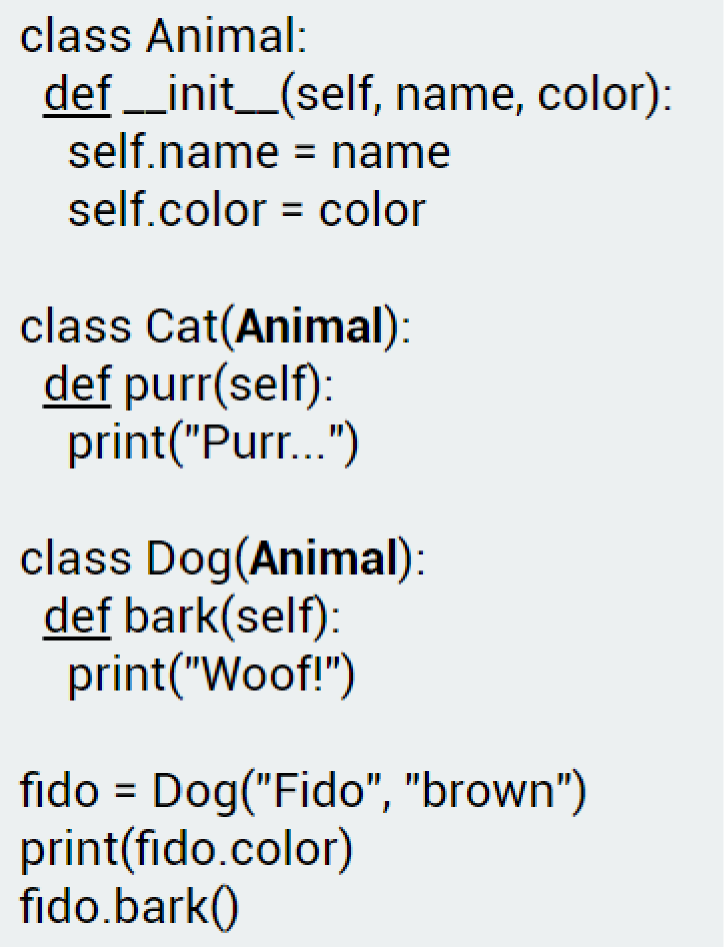
```
class Cat:
  def __init__(self, color, legs):
    self.color = color
    self.legs = legs

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



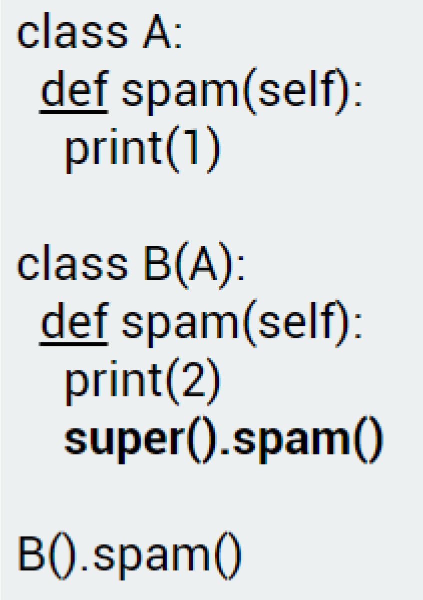
```
class Dog:
  def __init__(self, name, color):
    self.name = name
    self.color = color

  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()
```

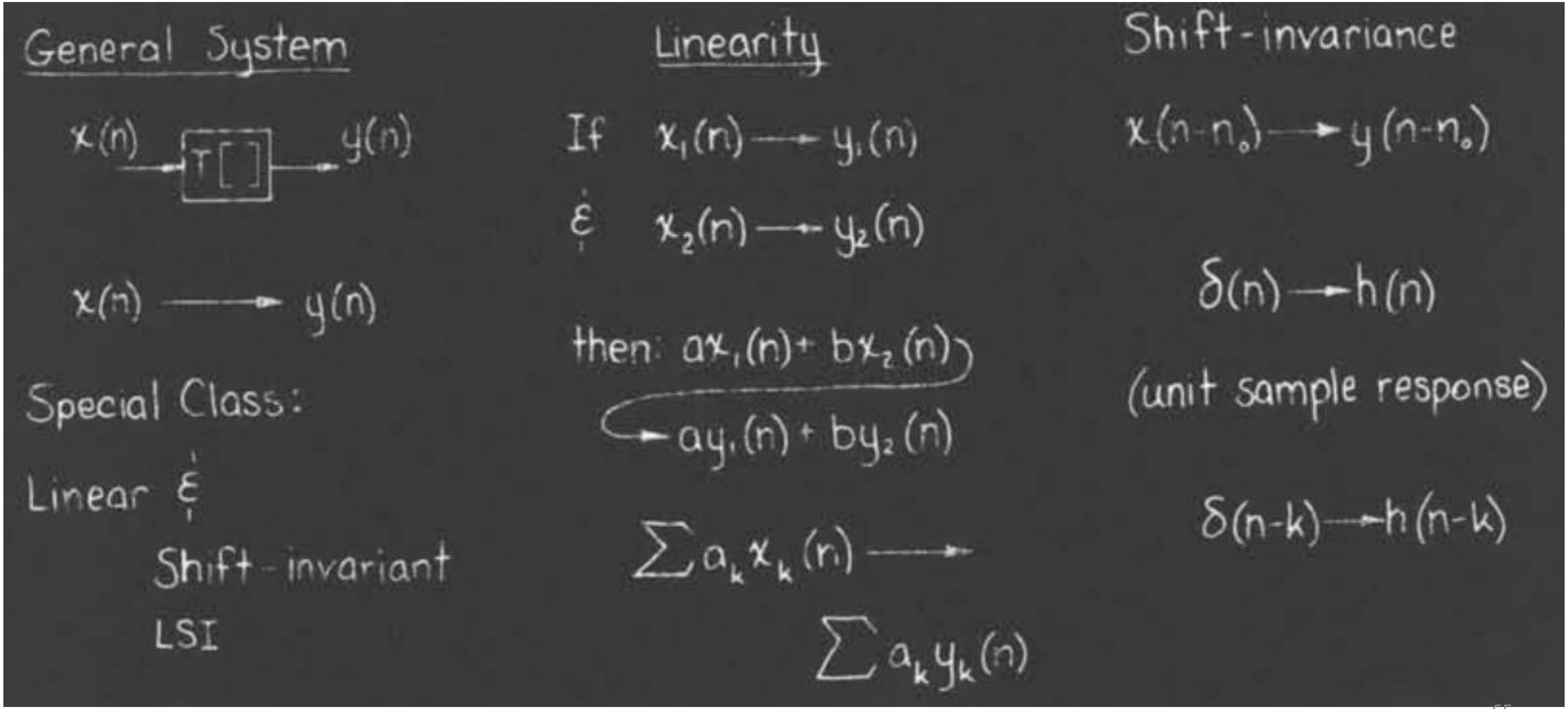


```
class A:
  def spam(self):
    print(1)

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

$$x(n) \longrightarrow \boxed{T[\;]} \longrightarrow y(n)$$

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

Special Class:

Linear &

Shift-invariant

LSI

Linearity

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

& $x_2(n) \longrightarrow y_2(n)$

then: $a x_1(n) + b x_2(n) \longrightarrow a y_1(n) + b y_2(n)$

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

Shift-invariance

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

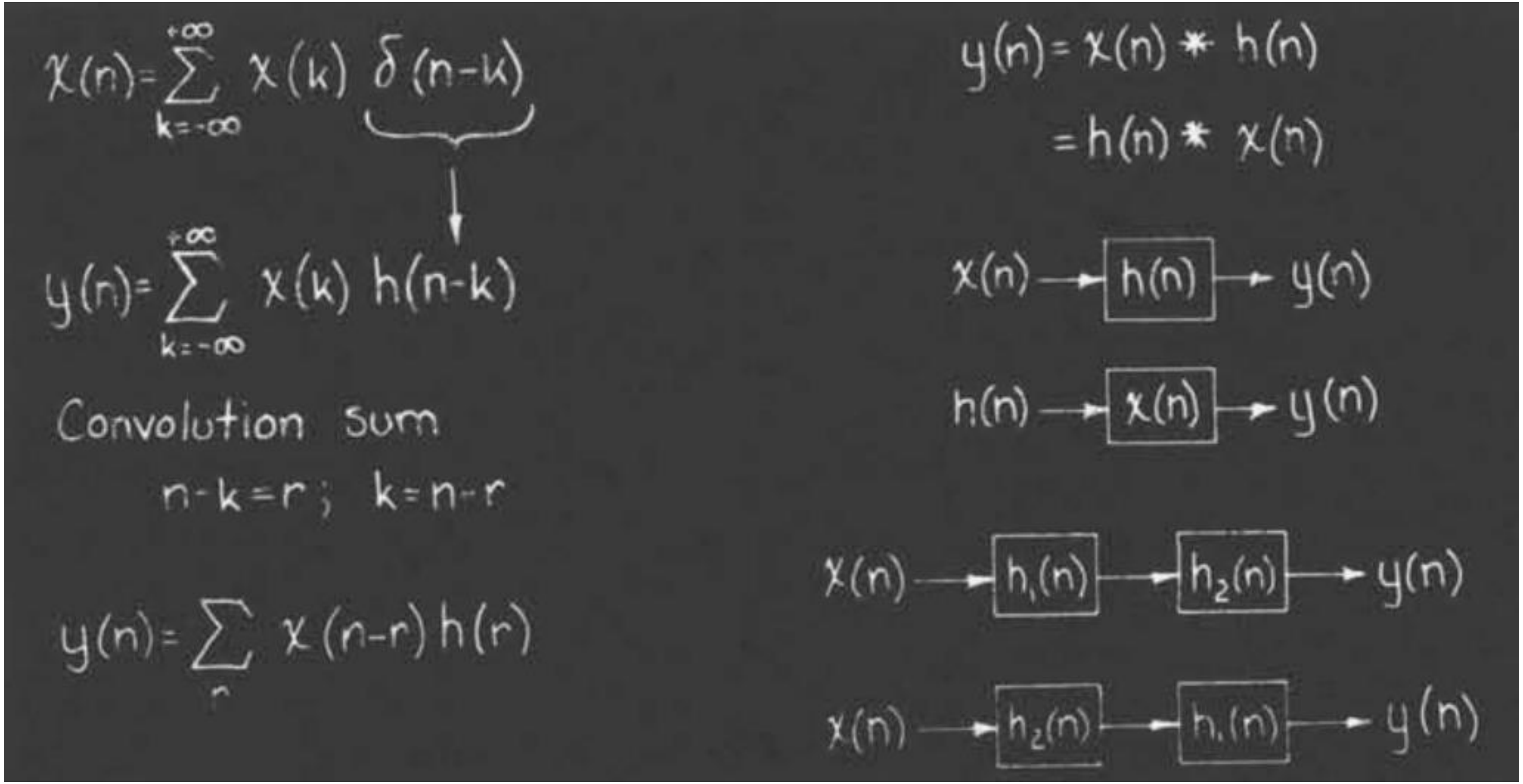
$$\delta(n) \longrightarrow h(n)$$

(unit sample response)

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

MIT DSP Course

Class Review



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

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

Convolution sum

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

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

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

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

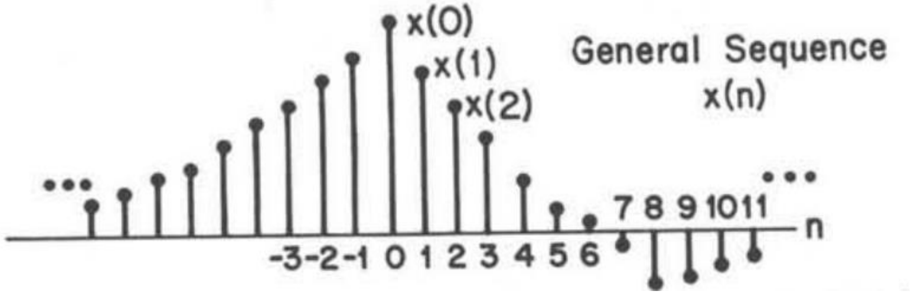
$$x(n) \longrightarrow \boxed{h(n)} \longrightarrow y(n)$$

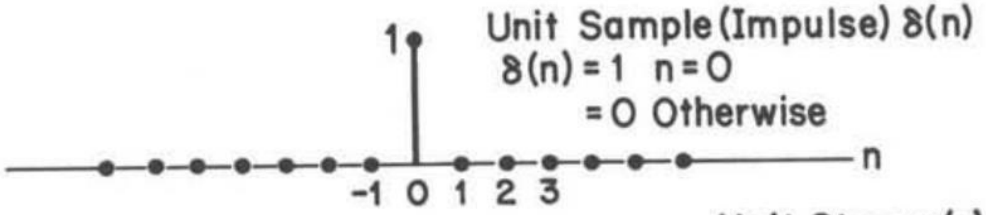
$$h(n) \longrightarrow \boxed{x(n)} \longrightarrow y(n)$$

$$x(n) \longrightarrow \boxed{h_1(n)} \longrightarrow \boxed{h_2(n)} \longrightarrow y(n)$$

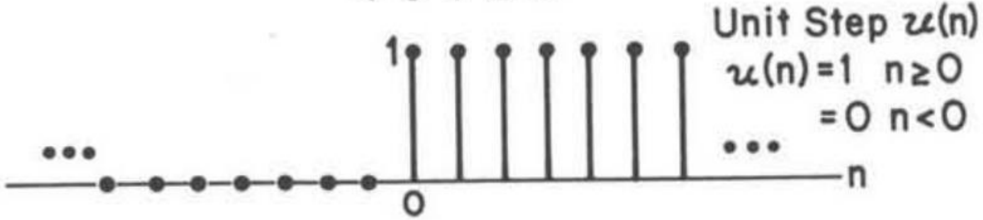
$$x(n) \longrightarrow \boxed{h_2(n)} \longrightarrow \boxed{h_1(n)} \longrightarrow y(n)$$

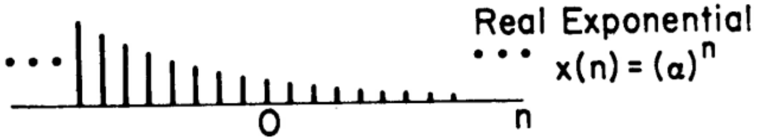
Class Review

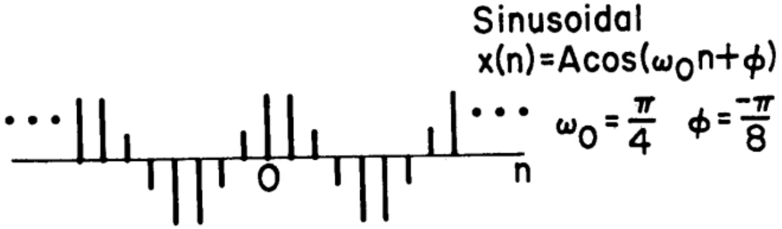




1
Unit Sample (Impulse) $\delta(n)$
$\delta(n) = 1 \quad n = 0$
$= 0$ Otherwise
-1 0 1 2 3
n

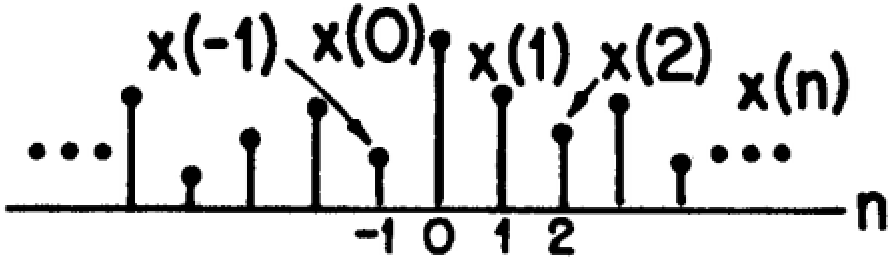






$\delta(n) = u(n) - u(n-1)$

Class Review



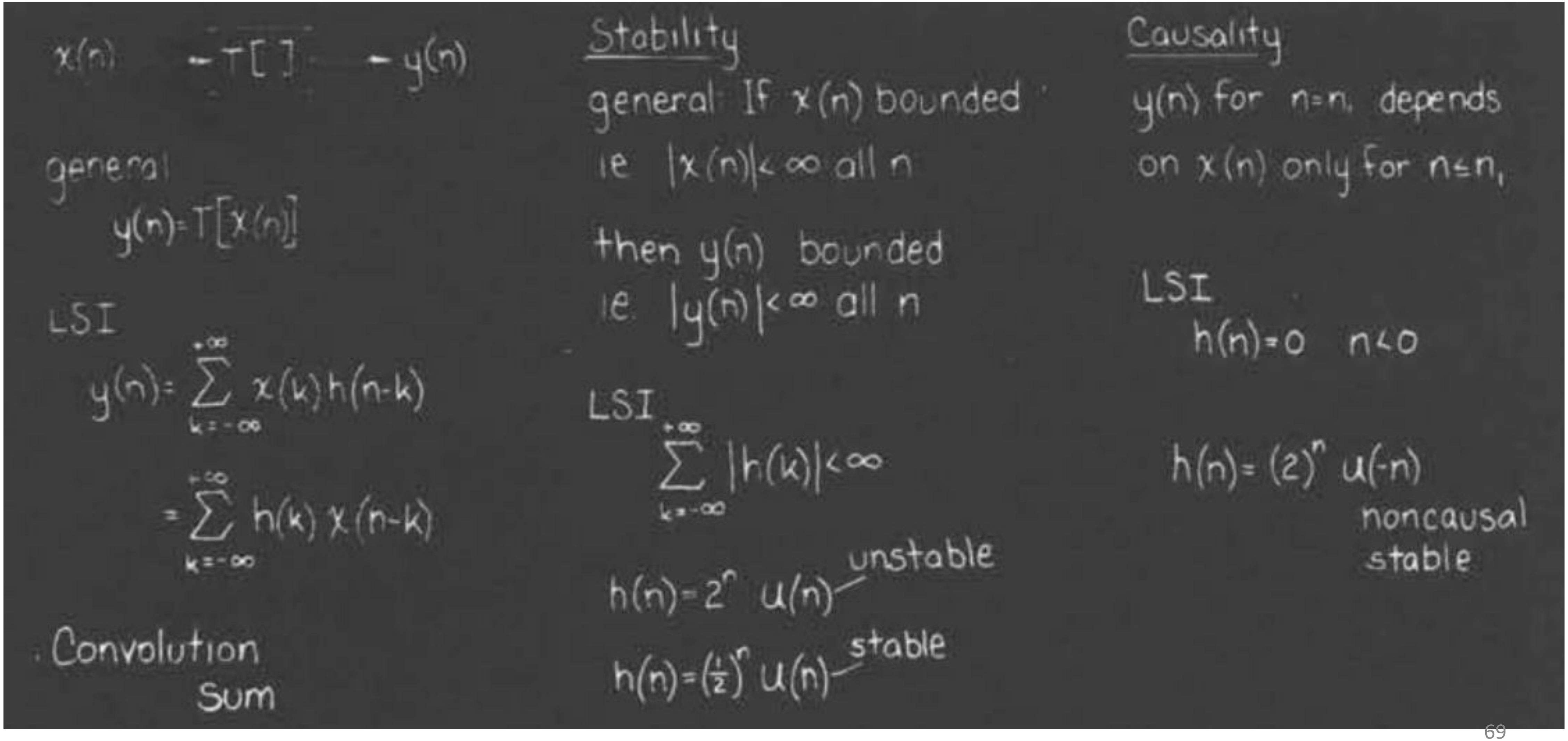
$$x(n) =$$

$$x(0)\delta(n) + x(1)\delta(n-1)$$

$$+ x(-1)\delta(n+1) + \cdots$$

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

Class Review



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

general

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

LSI

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

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

Convolution Sum

Stability

general: If $x(n)$ bounded

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

then $y(n)$ bounded

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

LSI

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

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

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

Causality

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

LSI

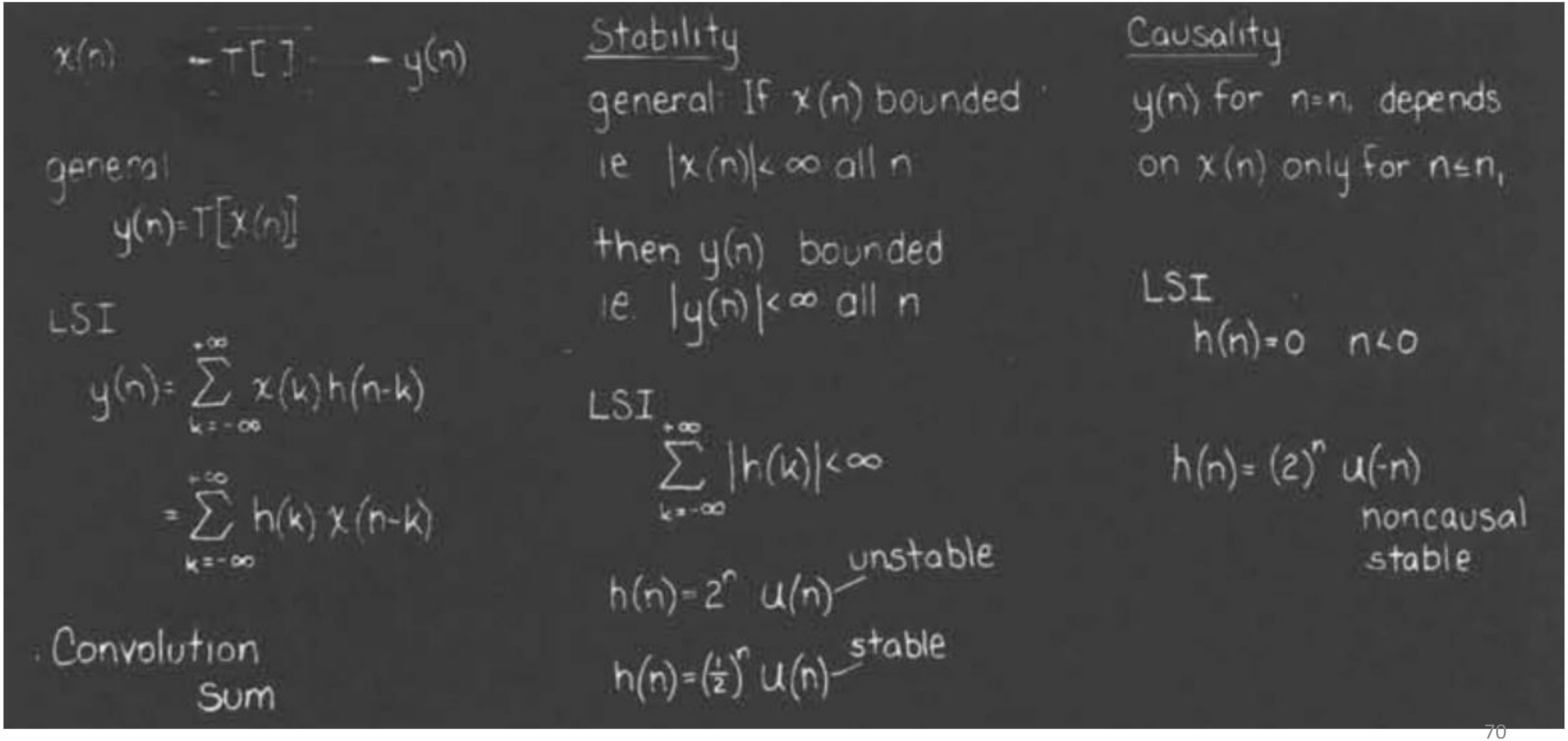
$h(n) = 0 \quad n < 0$

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

noncausal

stable

Class Review



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

general

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

LSI

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

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

Convolution Sum

Stability

general: If $x(n)$ bounded

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

then $y(n)$ bounded

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

LSI

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

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

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

Causality

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

LSI

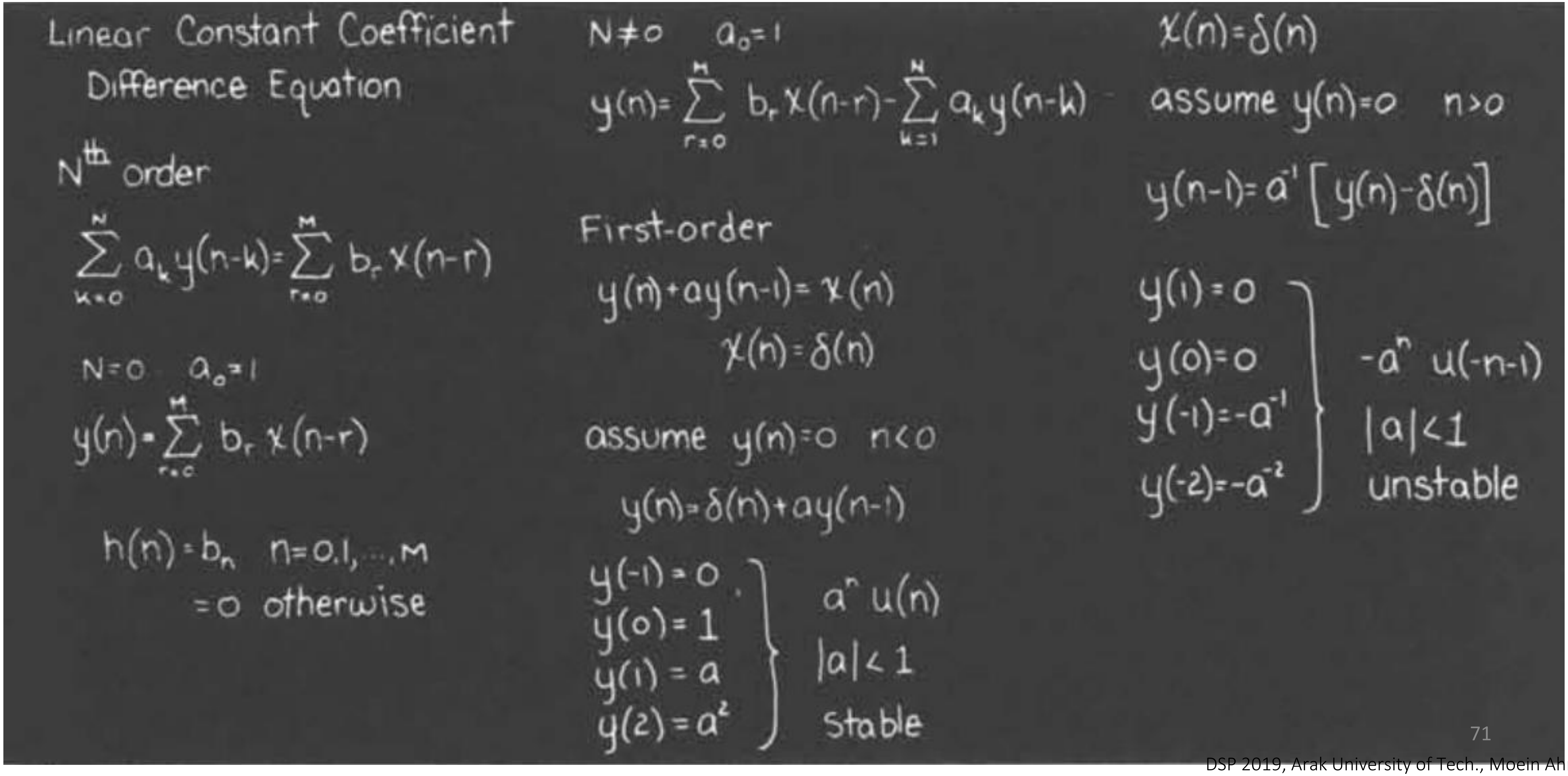
$h(n)=0 \quad n<0$

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

noncausal

stable

Class Review



Linear Constant Coefficient Difference Equation

N^{th} order

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

$N=0 \quad a_0=1$

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

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

$N \neq 0 \quad a_0=1$

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

First-order

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

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

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

$$\left.\begin{aligned} y(-1)&=0 \\ y(0)&=1 \\ y(1)&=a \\ y(2)&=a^2 \end{aligned}\right\} \quad \begin{aligned} &a^n u(n) \\ &|a|<1 \\ &\text{Stable} \end{aligned}$$

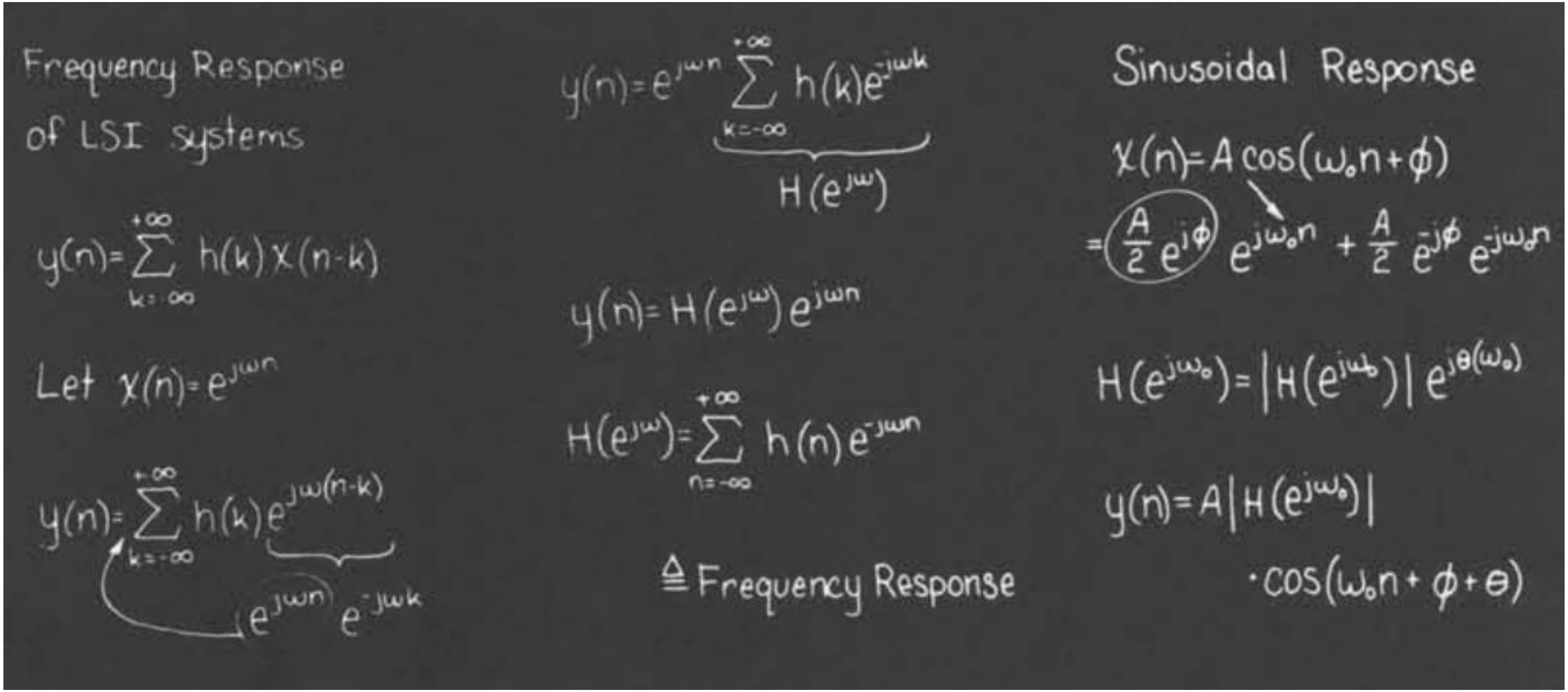
$$x(n) = \delta(n)$$

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

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

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

Class Review



Frequency Response of LSI systems

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

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

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

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

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

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

≜ Frequency Response

Sinusoidal Response

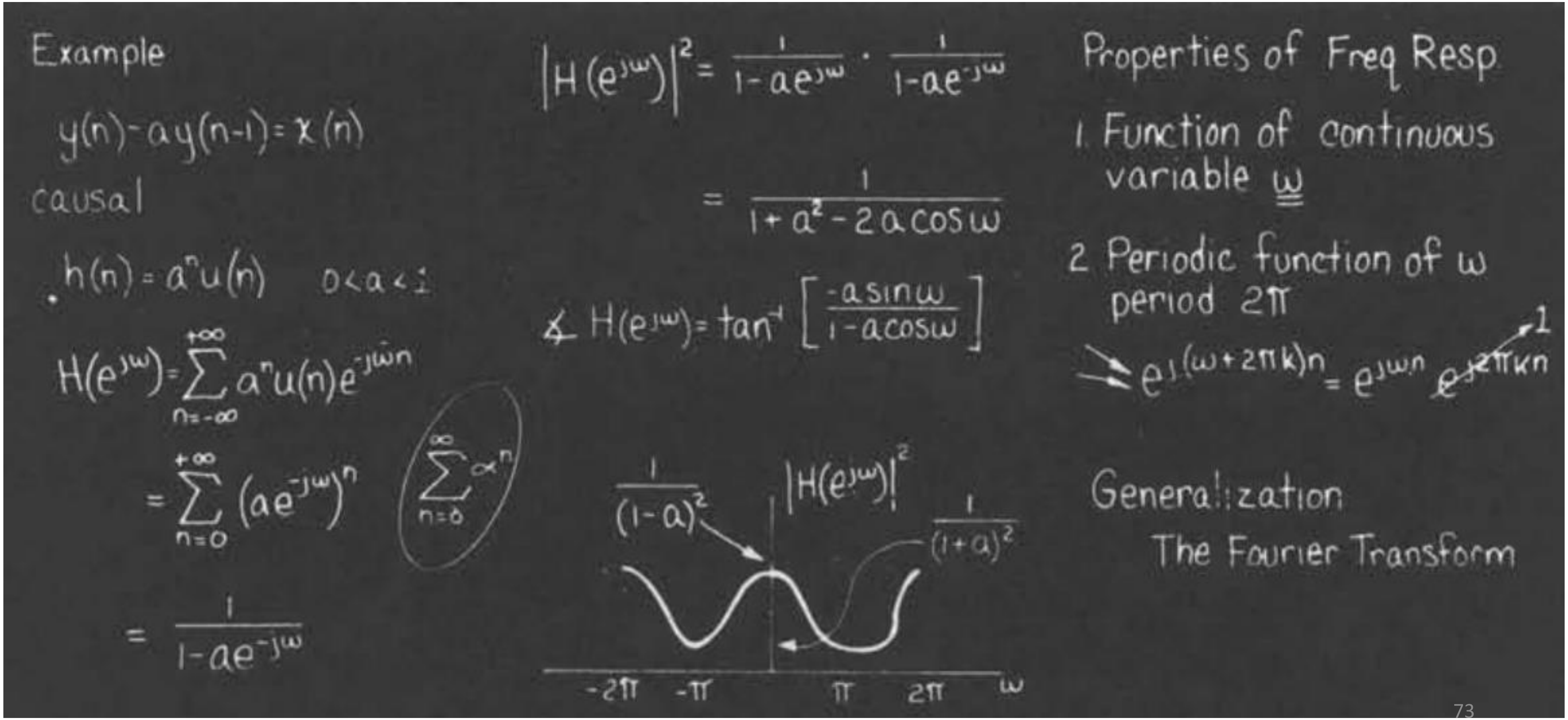
$$x(n)=A\cos(\omega_0 n+\phi)$$

$$=\boxed{\frac{A}{2}e^{j\phi}}\,e^{j\omega_0 n}+\frac{A}{2}e^{-j\phi}\,e^{-j\omega_0 n}$$

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

$$y(n)=A\left|H(e^{j\omega_0})\right|\cdot\cos(\omega_0 n+\phi+\theta)$$

Class Review



Example

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

causal

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

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

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

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

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

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

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

$\frac{1}{(1-a)^2}$
$|H(e^{j\omega})|^2$
$\frac{1}{(1+a)^2}$
-2π
$-\pi$
π
2π
ω

Properties of Freq Resp

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

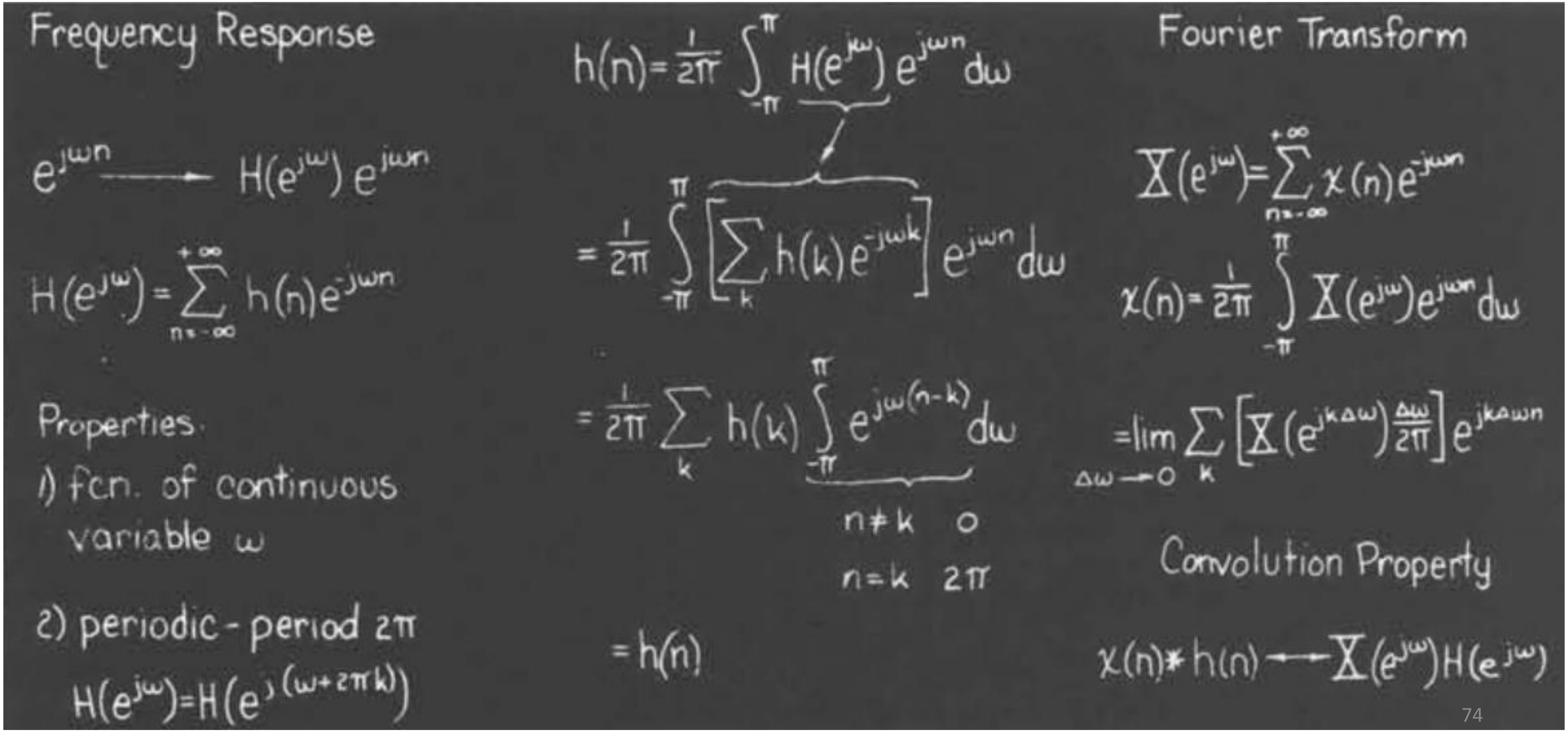
2. Periodic function of ω period 2π

$$e^{j(\omega+2\pi k)n} = e^{j\omega n}\,\overset{1}{\cancel{e^{j2\pi kn}}}$$

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 ω

2) periodic - period 2π

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

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

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

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

$n \neq k \quad 0$

$n = k \quad 2\pi$

$$= h(n)$$

Fourier Transform

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

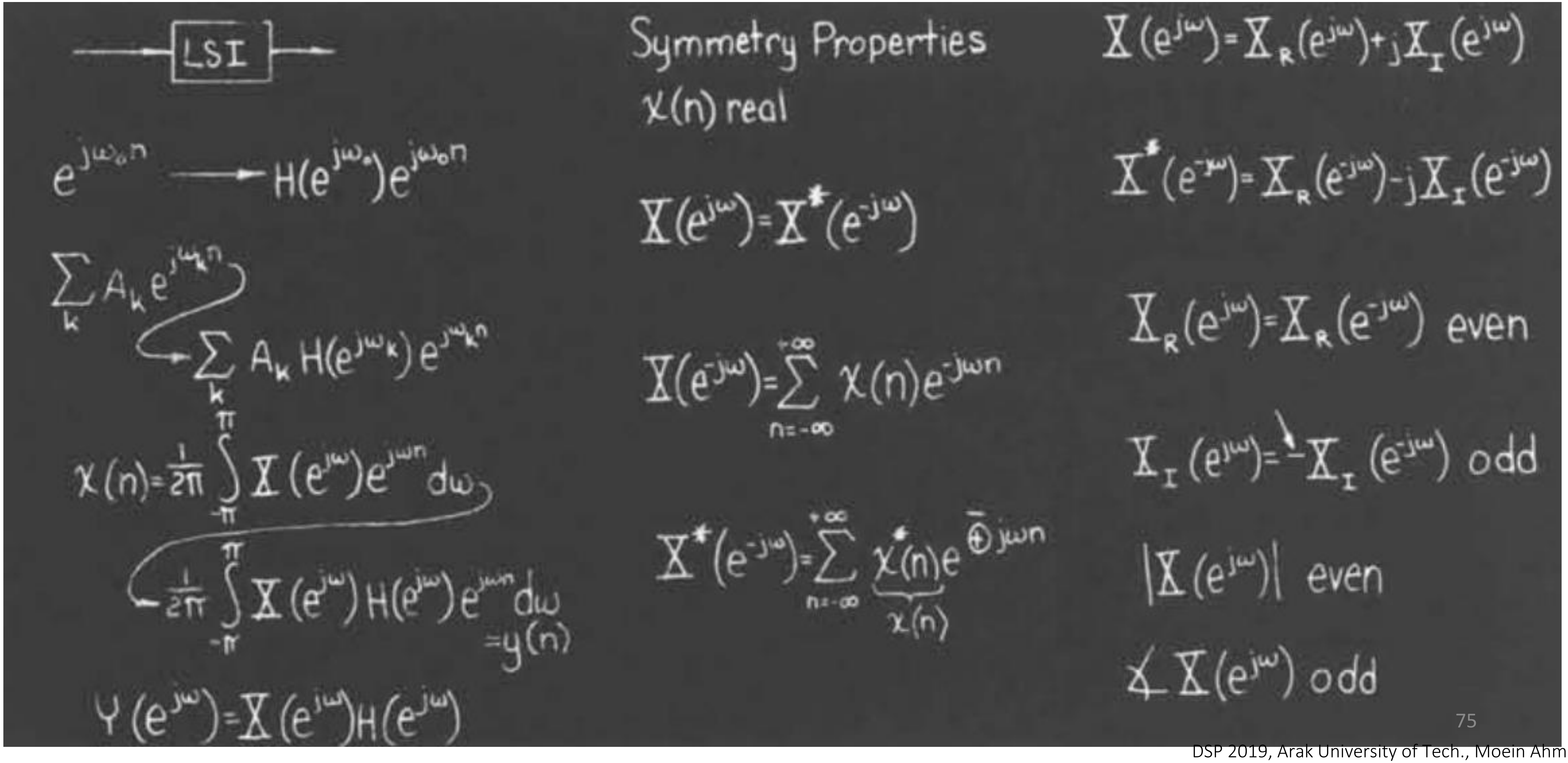
$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega n}\, d\omega$$

$$= \lim_{\Delta\omega \to 0} \sum_k \left[X(e^{jk\Delta\omega})\frac{\Delta\omega}{2\pi}\right] e^{jk\Delta\omega n}$$

Convolution Property

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

Class Review



$$\longrightarrow \boxed{LSI} \longrightarrow$$

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

$$\sum_k A_k e^{j\omega_k n}$$

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

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

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

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

Symmetry Properties

$x(n)$ real

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

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

$$X^*(e^{-j\omega}) = \sum_{n=-\infty}^{+\infty} \underbrace{x^*(n)}_{x(n)} e^{\overset{-}{\oplus} j\omega n}$$

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

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

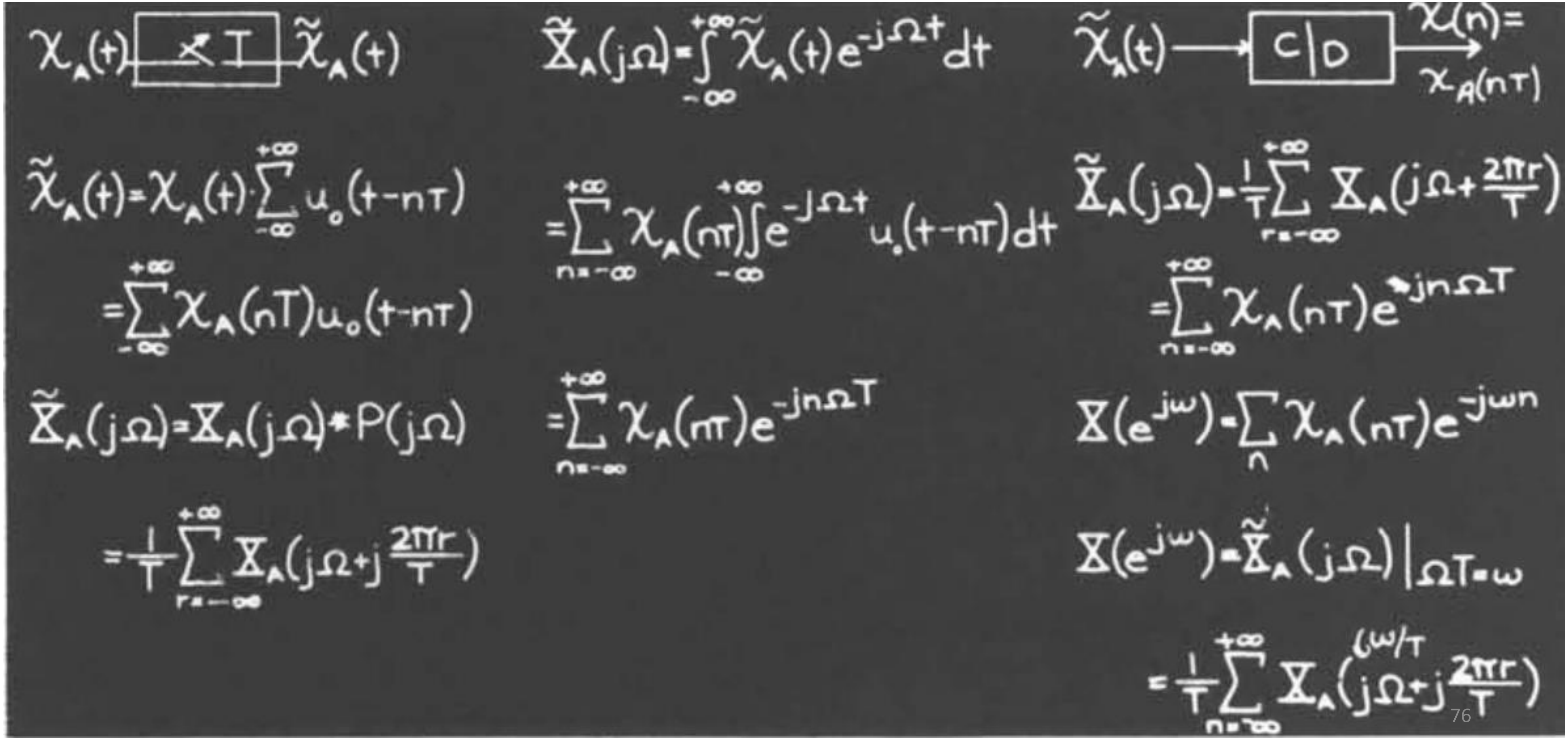
$X_R(e^{j\omega}) = X_R(e^{-j\omega})$ even

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

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

$\measuredangle X(e^{j\omega})$ odd

Class Review



$$x_A(t) \to \boxed{\times T} \to \tilde{x}_A(t)$$

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

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

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

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

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

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

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

$$\tilde{x}_A(t) \longrightarrow \boxed{C|D} \xrightarrow[x_A(nT)]{x(n)=}$$

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

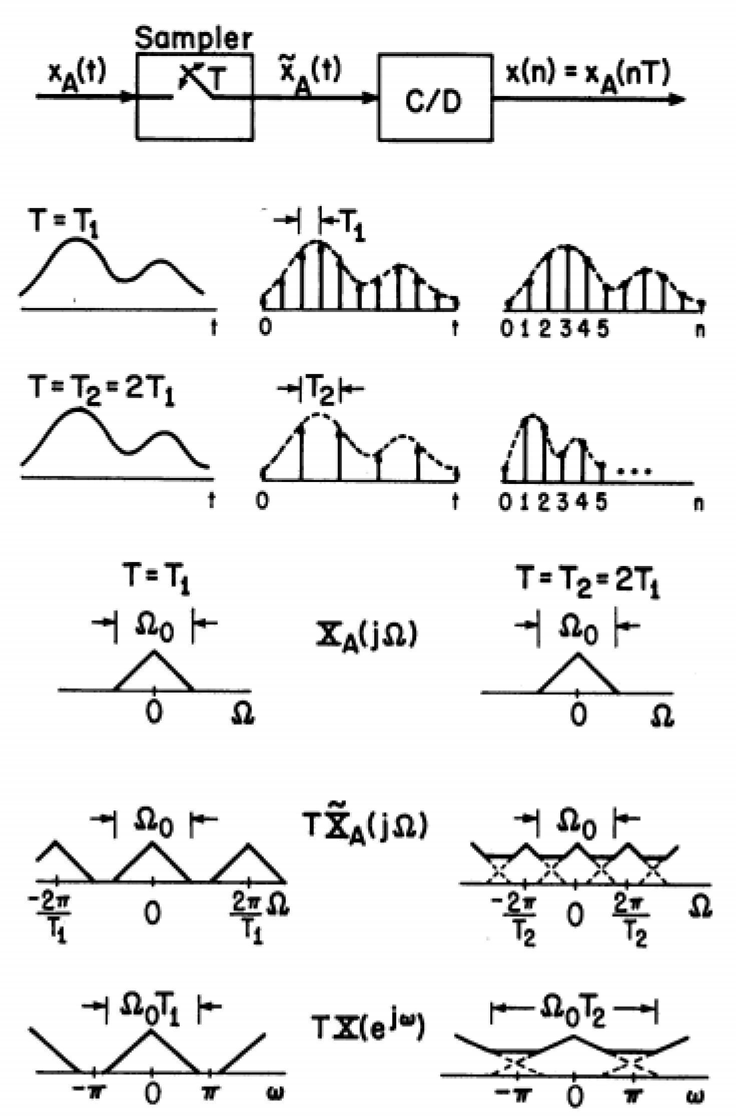
$$= \sum_{n=-\infty}^{+\infty} x_A(nT)\,e^{-jn\Omega T}$$

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

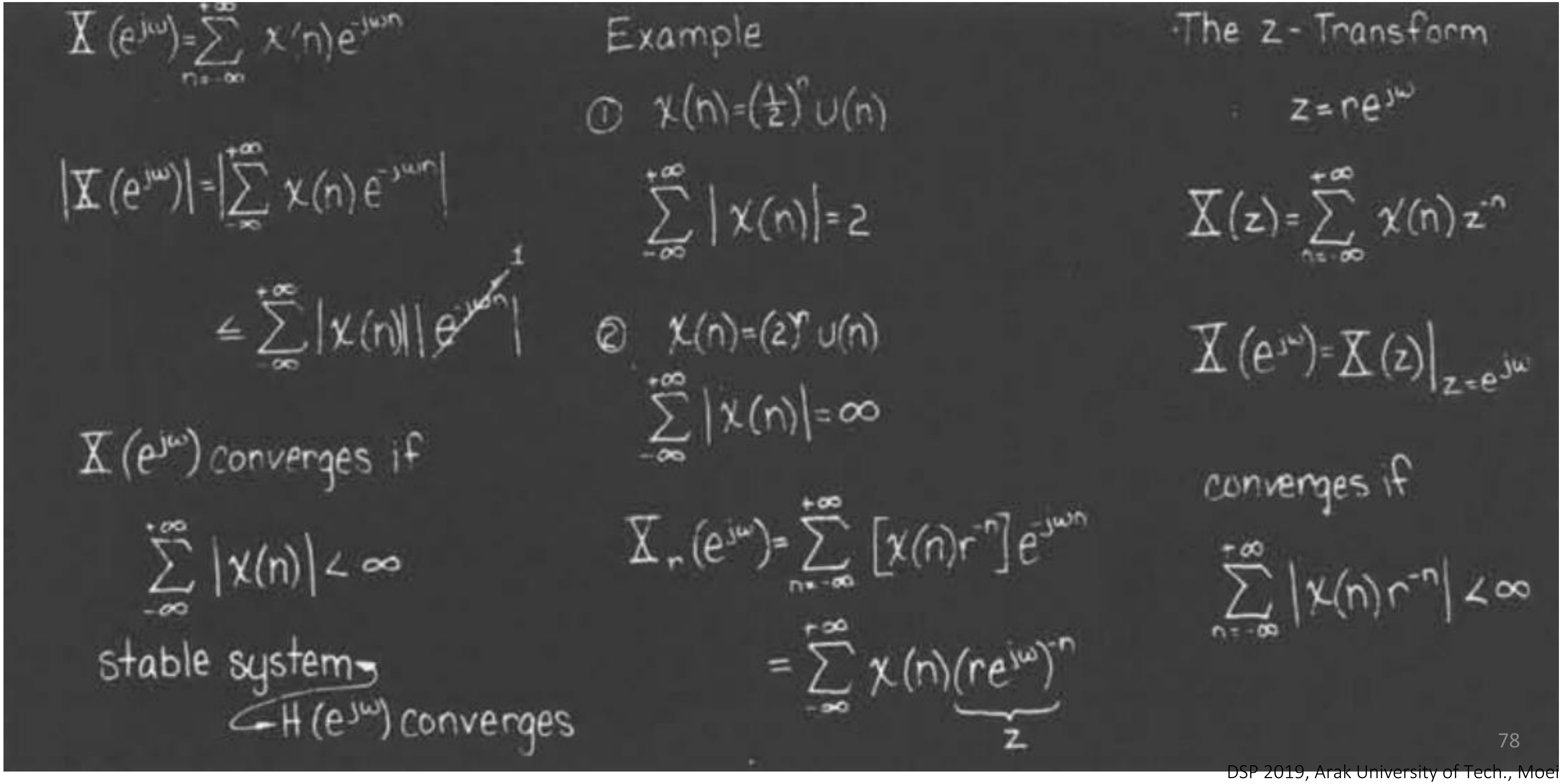
$$X(e^{j\omega}) = \tilde{X}_A(j\Omega)\Big|_{\Omega T=\omega}$$

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

Class Review



Class Review



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

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

$$\le \sum_{-\infty}^{+\infty} |x(n)| \left|\cancelto{1}{e^{-j\omega n}}\right|$$

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

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

stable system

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

Example

① $x(n) = (\frac{1}{2})^n u(n)$

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

② $x(n) = (2)^n u(n)$

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

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

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

The z-Transform

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

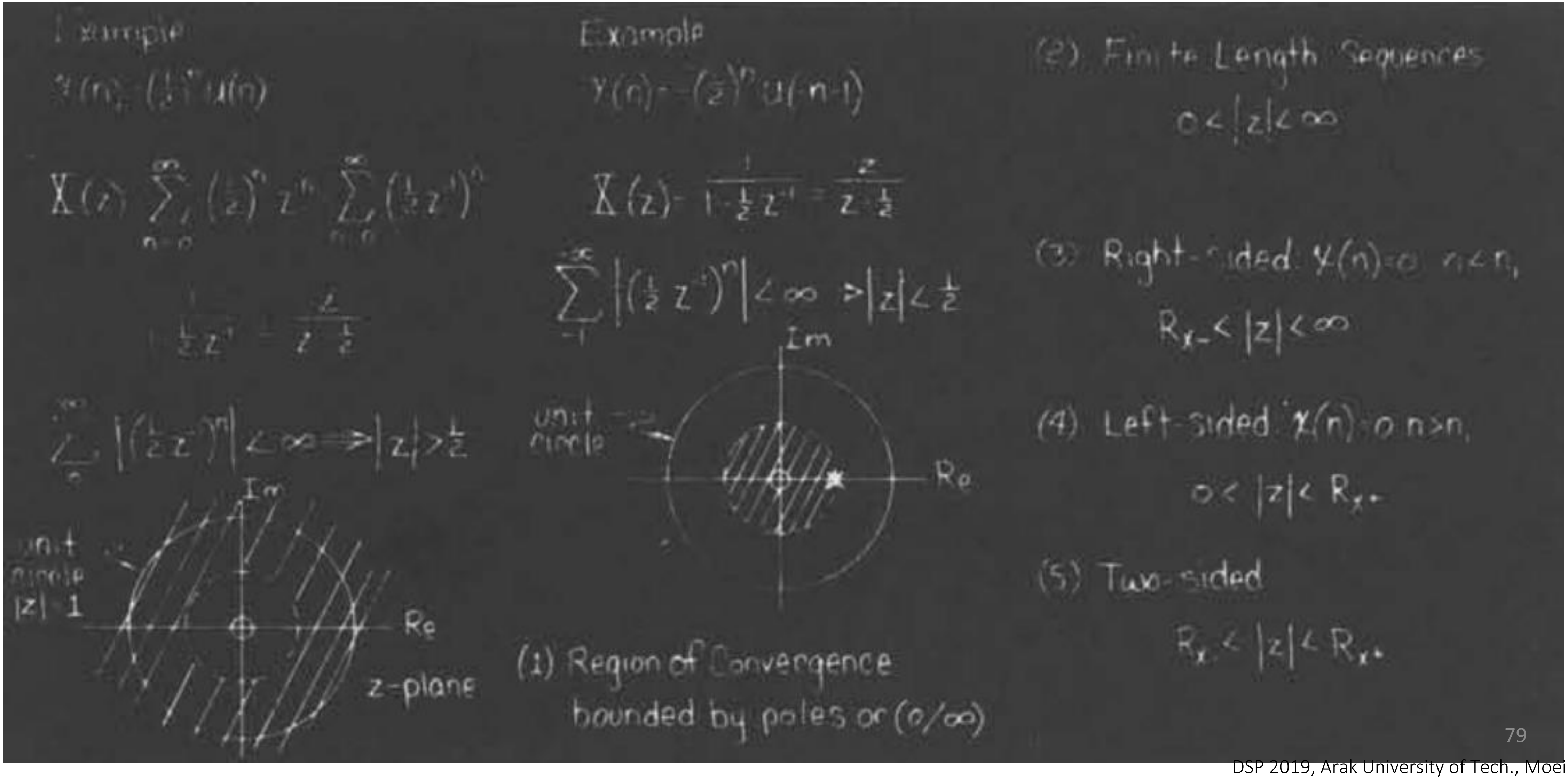
$$X(z) = \sum_{n=-\infty}^{+\infty} x(n) z^{-n}$$

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

converges if

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

Class Review



Example

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

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

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

$$\sum_{0}^{\infty} \left|\left(\frac{1}{2} z^{-1}\right)^n\right| < \infty \Rightarrow |z| > \frac{1}{2}$$

Example

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

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

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

(1) Region of Convergence bounded by poles or (0/∞)

(2) Finite Length Sequences

$0 < |z| < \infty$

(3) Right-sided $x(n) = 0 \quad n < n_1$

$R_{x-} < |z| < \infty$

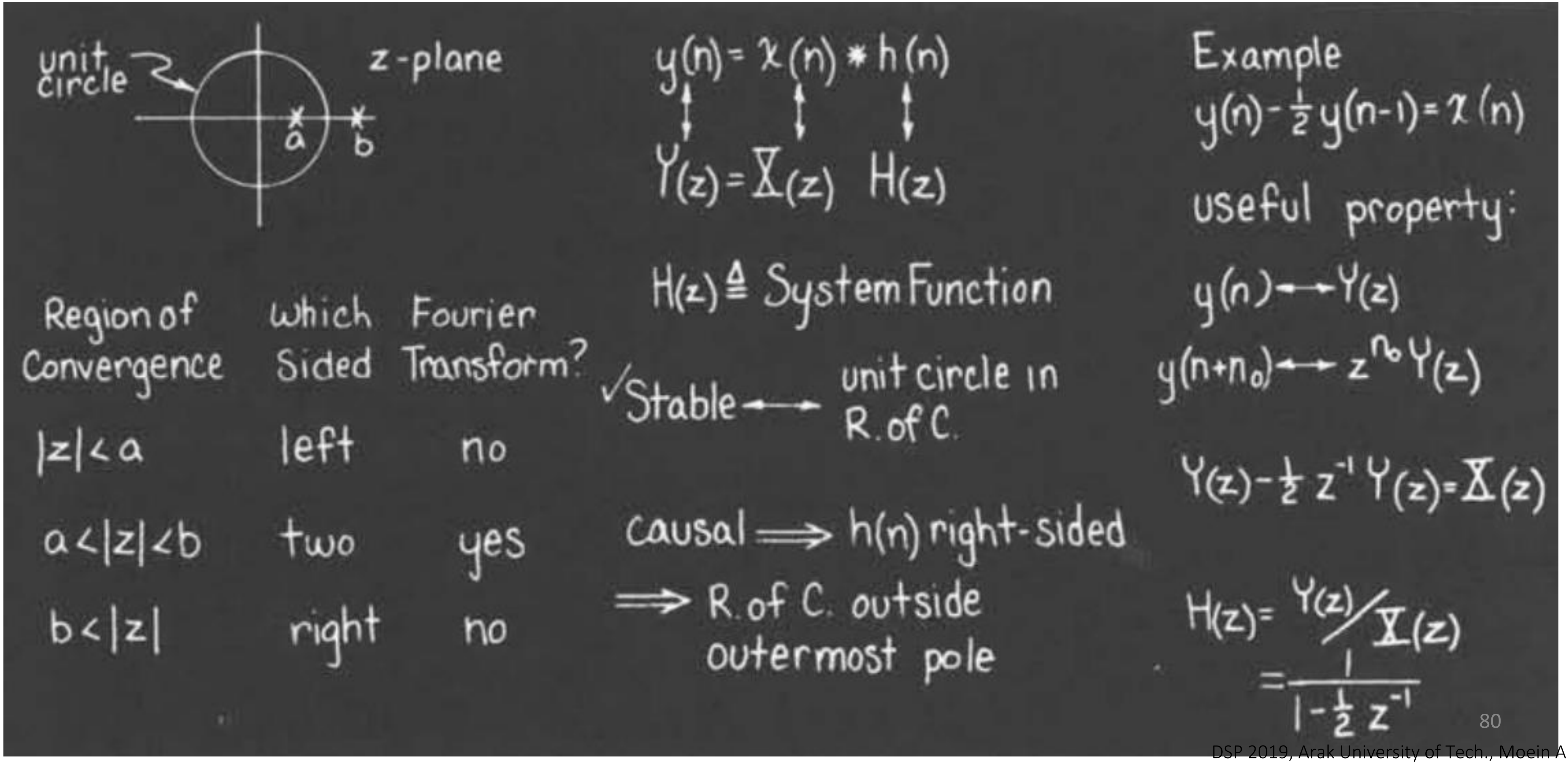
(4) Left-sided $x(n) = 0 \quad n > n_2$

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

(5) Two-sided

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

Class Review



unit circle
z-plane
a
b

Region of Convergence	Which Sided	Fourier Transform?
$\lvert z\rvert < a$	left	no
$a < \lvert z\rvert < b$	two	yes
$b < \lvert z\rvert$	right	no

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

$y(n) \updownarrow Y(z)$, $x(n) \updownarrow X(z)$, $h(n) \updownarrow H(z)$

$Y(z) = X(z)\ H(z)$

$H(z) \triangleq$ System Function

✓Stable ⟷ unit circle in R. of C.

causal $\Longrightarrow$ h(n) right-sided

$\Longrightarrow$ R. of C. outside outermost pole

Example

$y(n) - \frac{1}{2} y(n-1) = x(n)$

useful property:

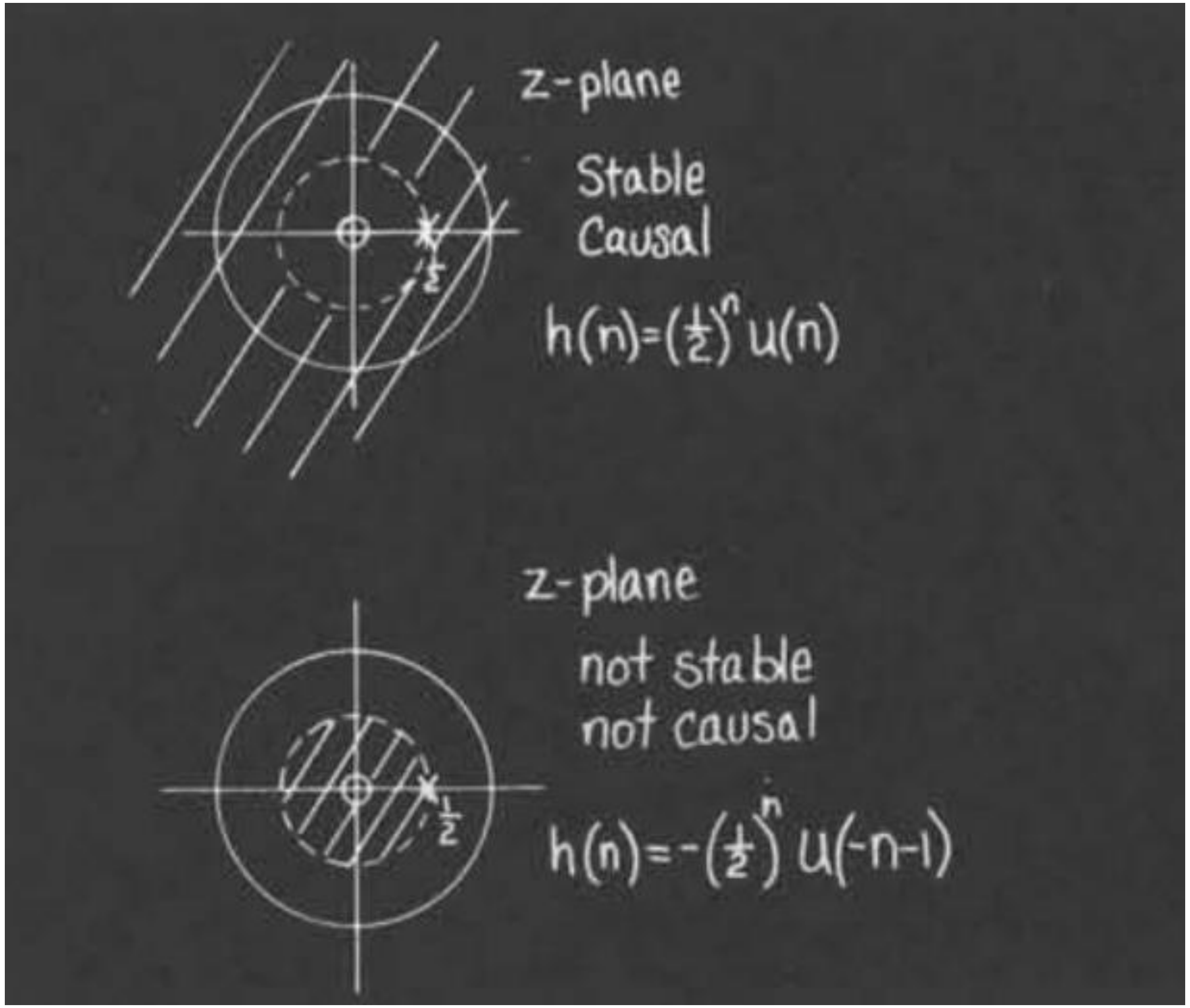
$y(n) \longleftrightarrow Y(z)$

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

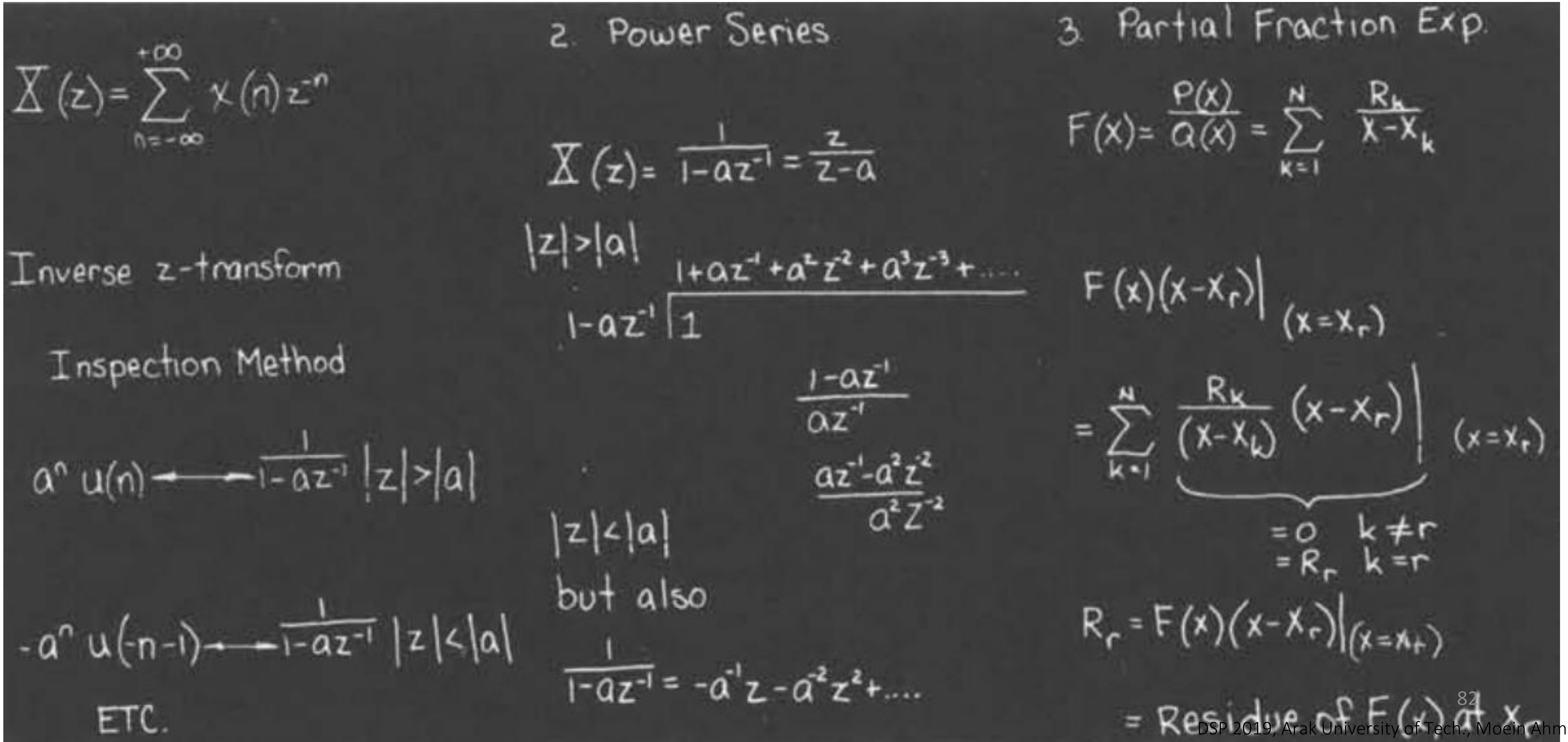
$Y(z) - \frac{1}{2} z^{-1} Y(z) = X(z)$

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

Class Review



Class Review



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

Inverse z-transform

Inspection Method

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

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

ETC.

2. Power Series

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

$|z|>|a|$

$$\begin{array}{r|l} & 1+az^{-1}+a^2z^{-2}+a^3z^{-3}+\ldots \\ \hline 1-az^{-1} & 1 \\ & \underline{1-az^{-1}} \\ & az^{-1} \\ & \underline{az^{-1}-a^2z^{-2}} \\ & a^2z^{-2} \end{array}$$

$|z|<|a|$

but also

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

3. Partial Fraction Exp.

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

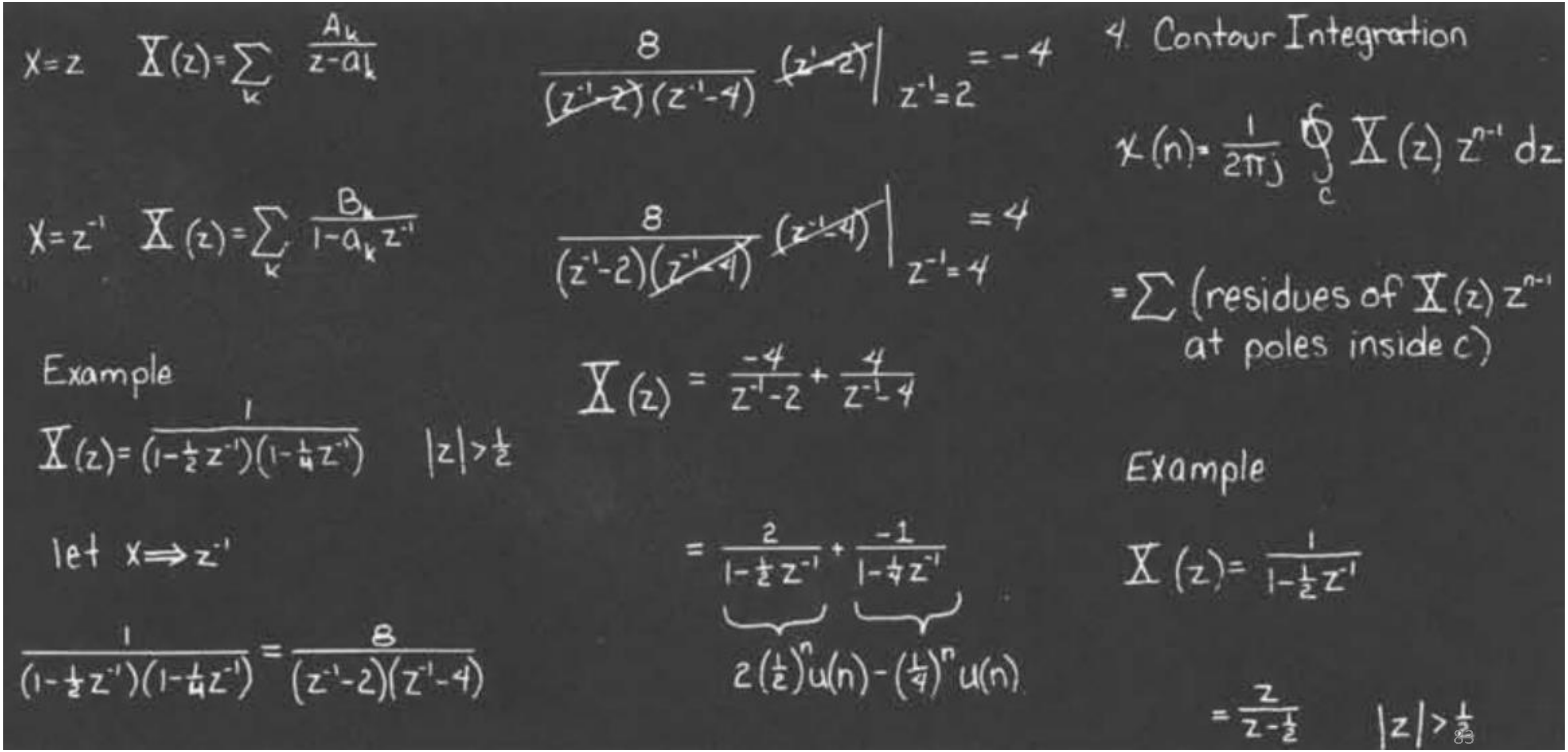
$$F(x)(x-x_r)\Big|_{(x=x_r)}$$

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

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

$= \text{Residue of } F(x) \text{ at } x_r$

Class Review



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

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

Example

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

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

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

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

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

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

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

4. Contour Integration

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

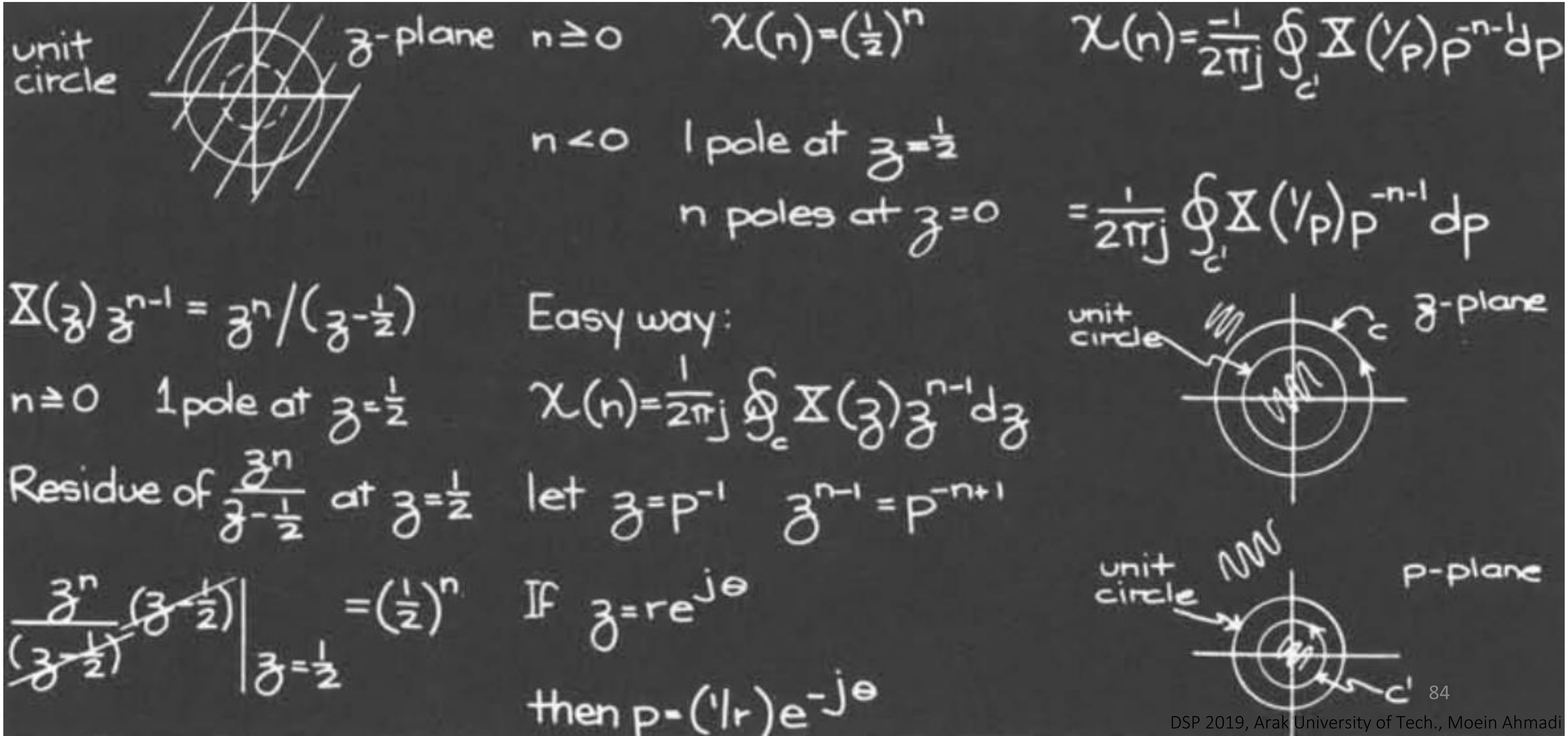
$= \sum$ (residues of $X(z)\, z^{n-1}$ at poles inside c)

Example

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

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

Class Review



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

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

n poles at $z = 0$

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

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

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

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

Easy way:

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

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

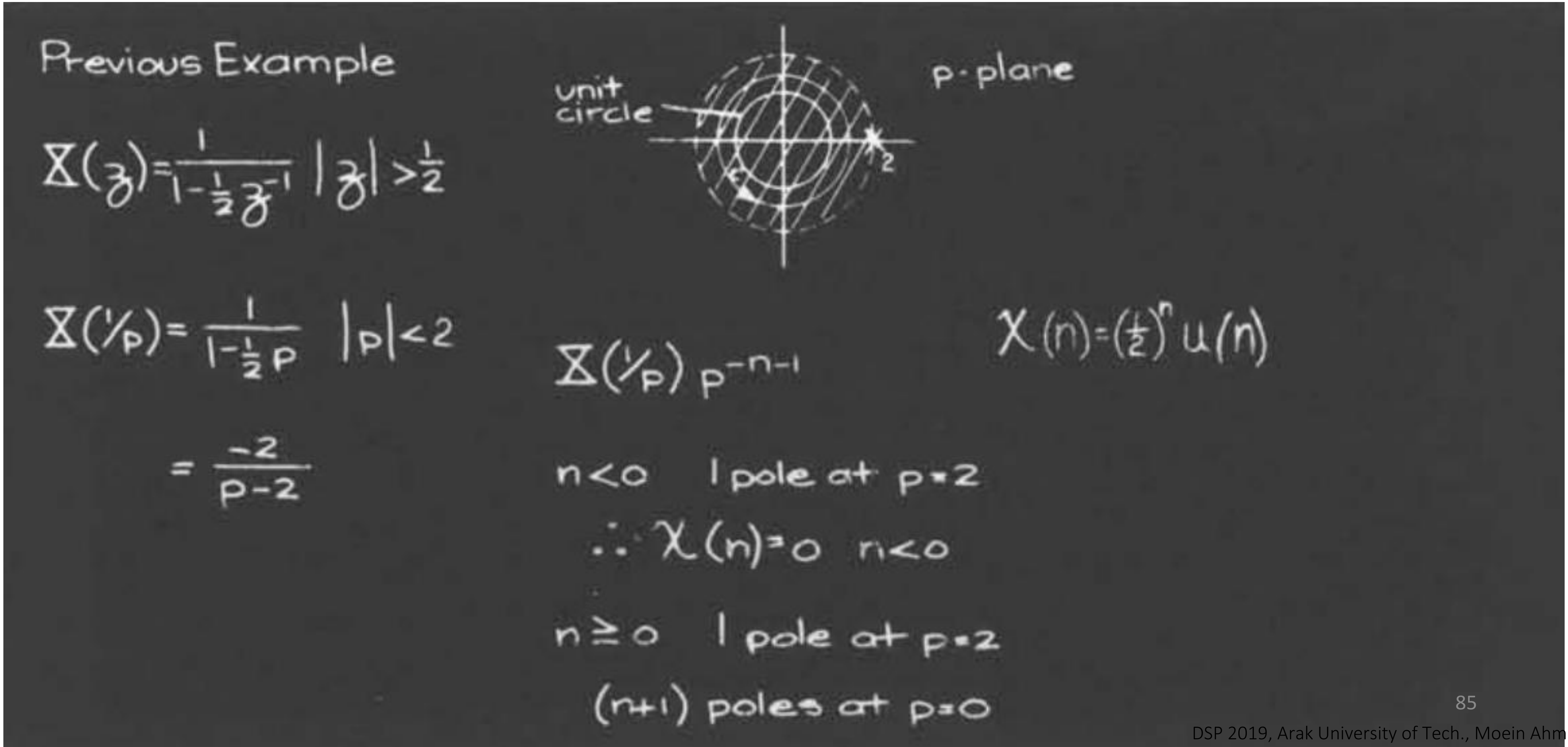
If $z = re^{j\theta}$

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

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

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

Class Review



Previous Example

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

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

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

unit circle
2
p-plane

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

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

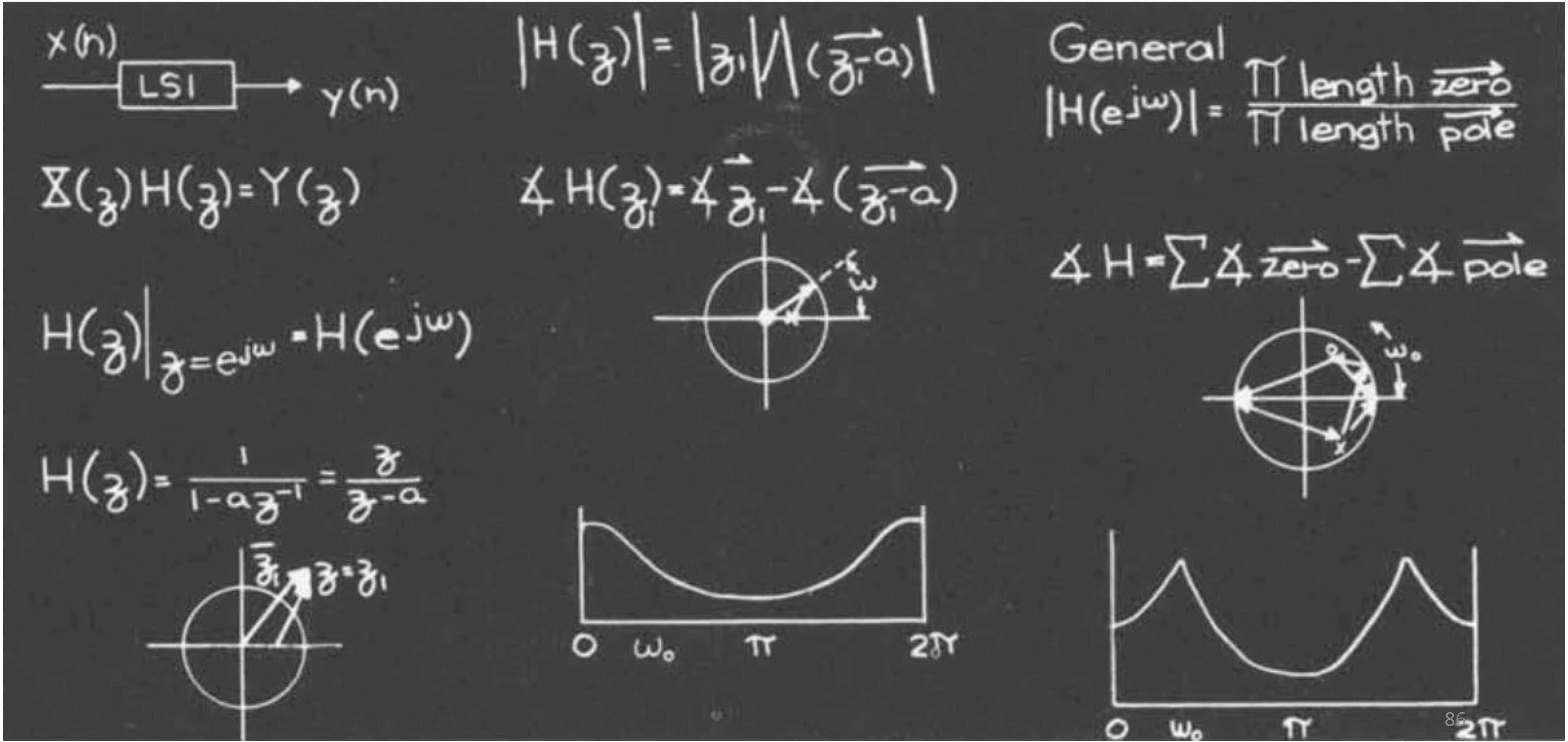
$\therefore\ x(n)=0 \quad n<0$

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

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

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

Class Review



$$x(n) \longrightarrow \boxed{\text{LSI}} \longrightarrow y(n)$$

$$X(z)H(z)=Y(z)$$

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

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

$$|H(z)| = |\vec{z_1}| / |(\overrightarrow{z_1-a})|$$

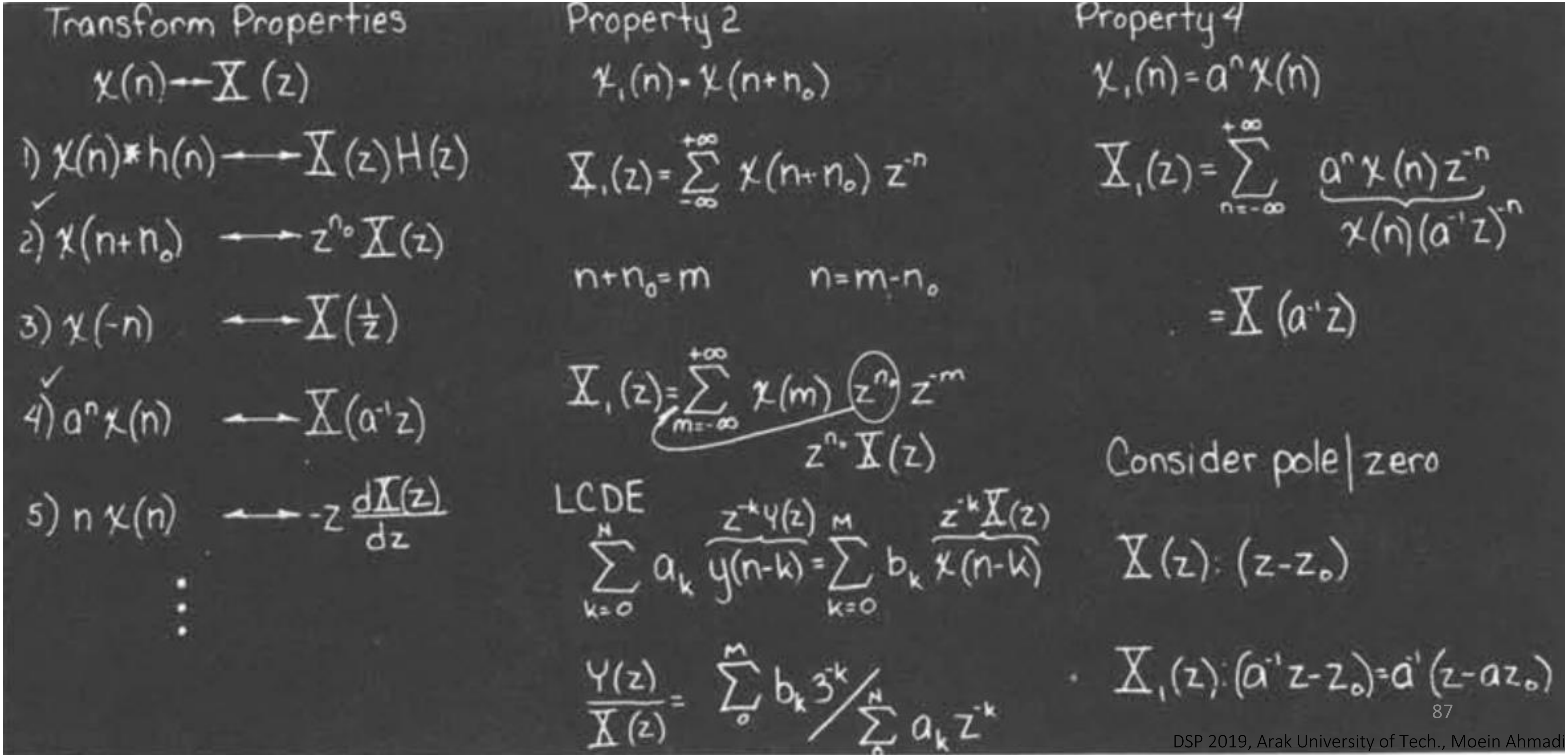
$$\measuredangle H(z_1) = \measuredangle \vec{z_1} - \measuredangle (\overrightarrow{z_1-a})$$

General

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

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

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

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

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

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

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

$z^{n_0}X(z)$

LCDE

$$\sum_{k=0}^{N} a_k \overbrace{y(n-k)}^{z^{-k}Y(z)} = \sum_{k=0}^{M} b_k \overbrace{x(n-k)}^{z^{-k}X(z)}$$

$$\frac{Y(z)}{X(z)} = \sum_{0}^{M} b_k z^{-k} \Big/ \sum_{0}^{N} a_k z^{-k}$$

Property 4

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

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

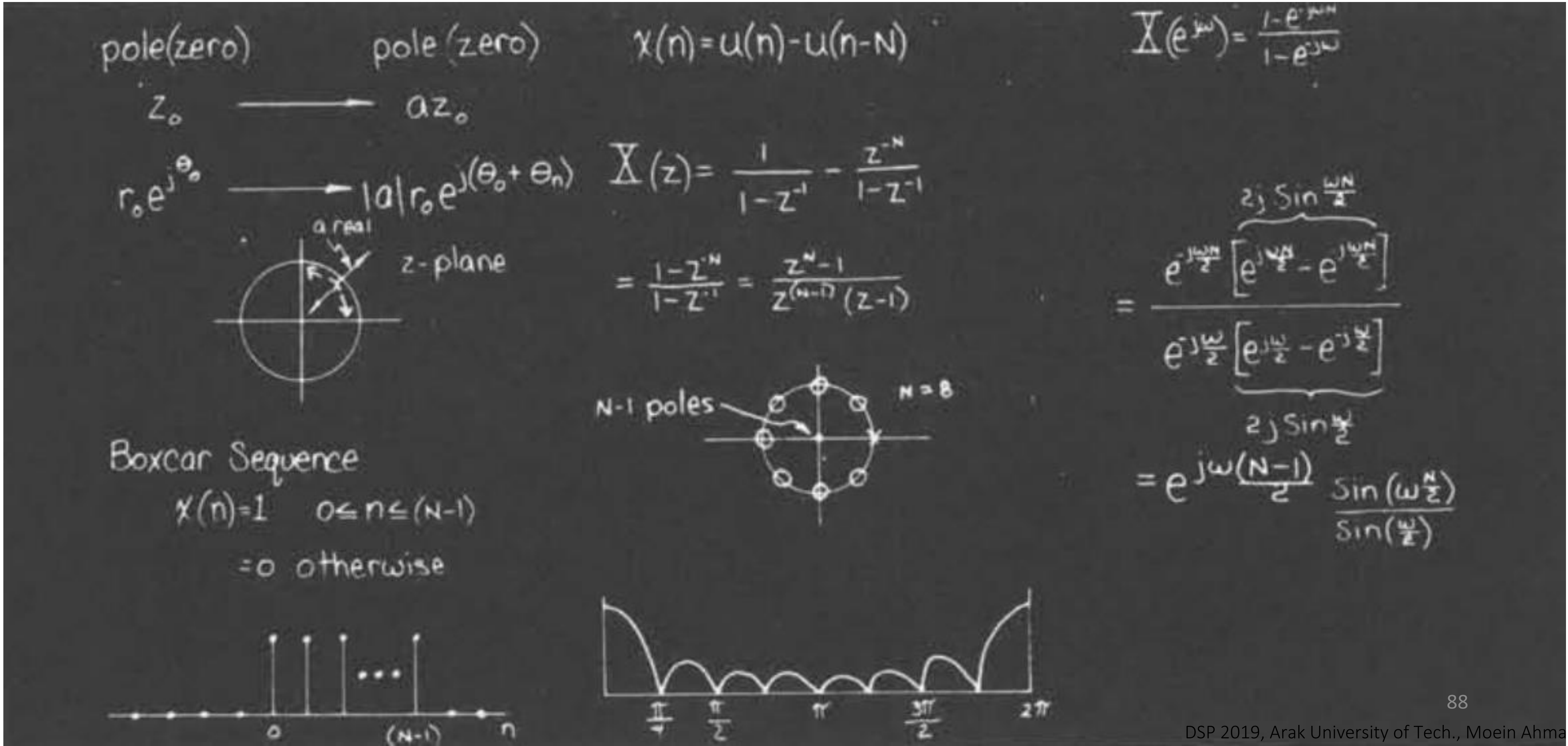
$$= X(a^{-1}z)$$

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) pole(zero)

$$z_0 \longrightarrow a z_0$$

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

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

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

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

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

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

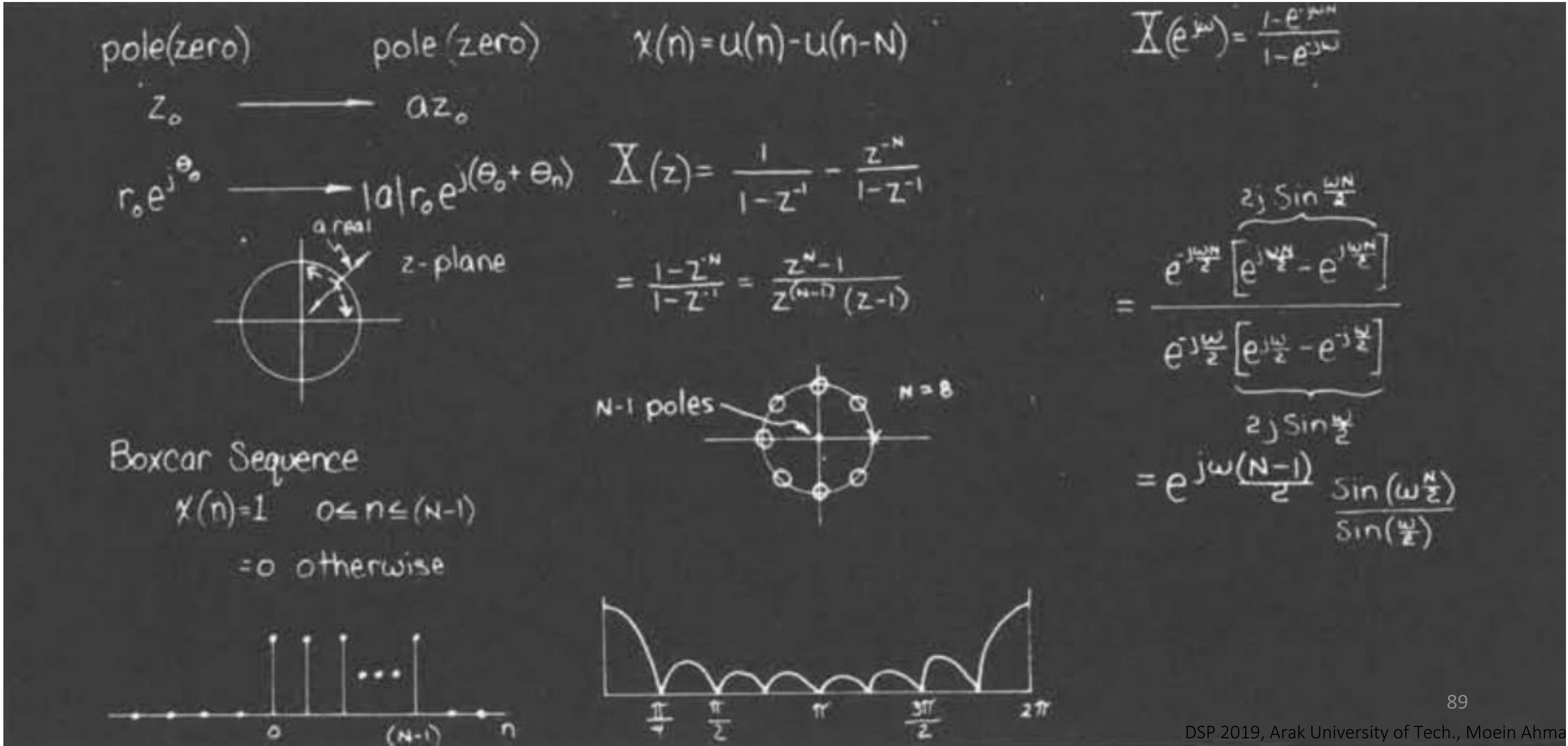
$$= e^{j\omega\frac{(N-1)}{2}} \frac{\sin(\omega\frac{N}{2})}{\sin(\frac{\omega}{2})}$$

Boxcar Sequence

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

$= 0$ otherwise

Class Review



pole(zero) $\longrightarrow$ pole(zero)

$z_0 \longrightarrow az_0$

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

a real

z-plane

Boxcar Sequence

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

$= 0$ otherwise

0

(N-1)

n

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

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

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

N-1 poles

N = 8

$\frac{\pi}{4}$

$\frac{\pi}{2}$

π

$\frac{3\pi}{2}$

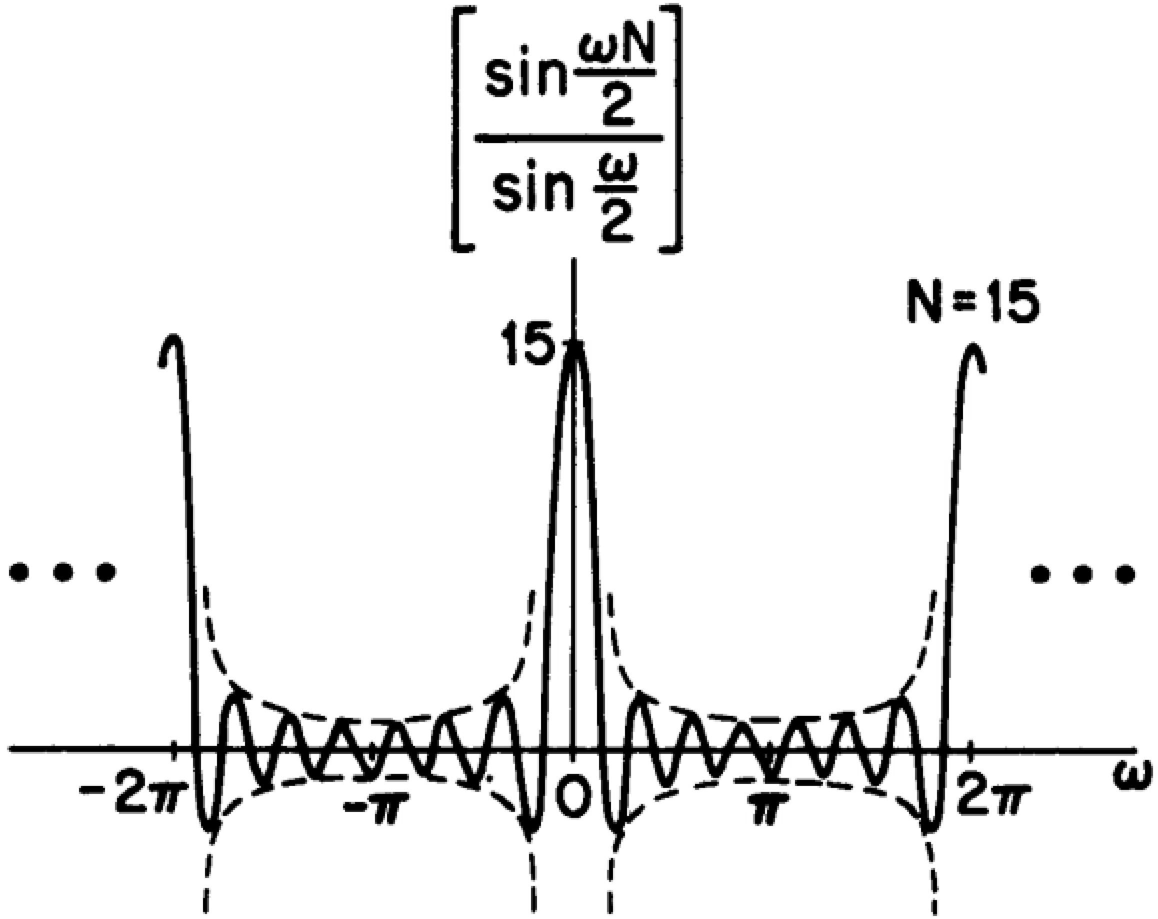
2π

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

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

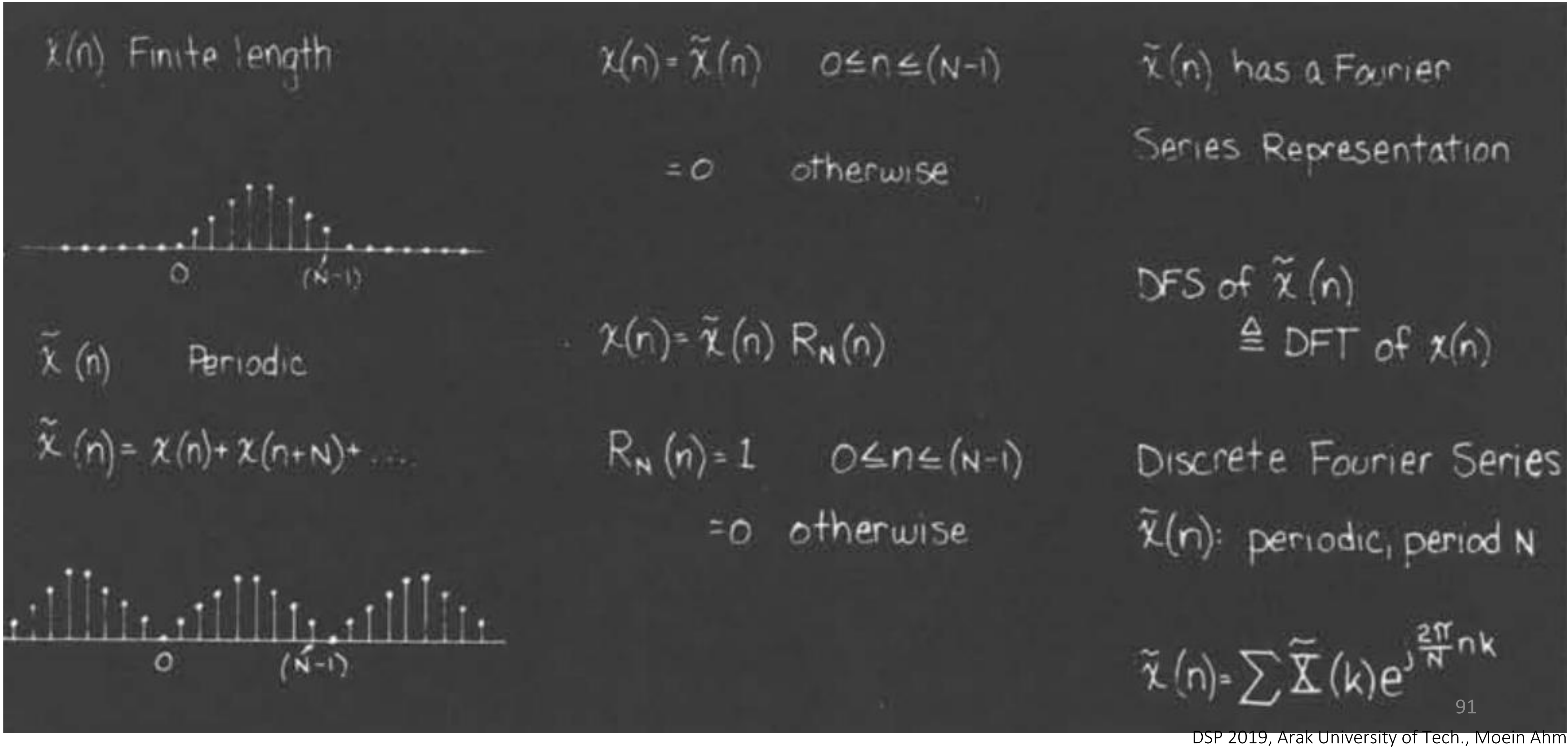
$$= e^{j\omega\frac{(N-1)}{2}} \frac{\sin(\omega\frac{N}{2})}{\sin(\frac{\omega}{2})}$$

Class Review



Fourier Transform of a rectangular sequence.

Class Review



$x(n)$ Finite length

$\tilde{x}(n)$ Periodic

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

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

$= 0 \quad$ otherwise

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

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

$= 0 \quad$ otherwise

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

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

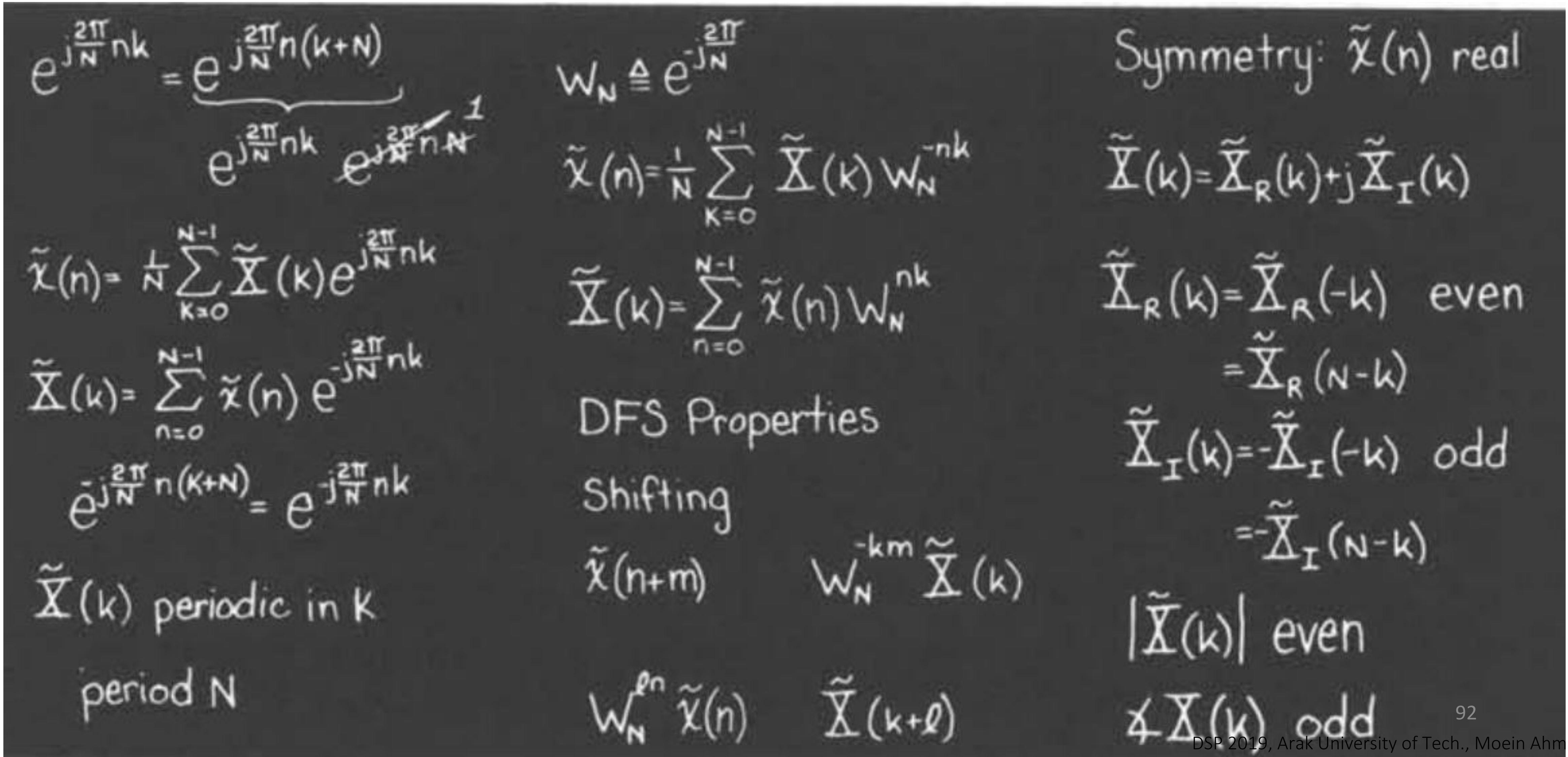
$\triangleq$ DFT of $x(n)$

Discrete Fourier Series

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

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

Class Review



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

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

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

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

$\tilde{X}(k)$ 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

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

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

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

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

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

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

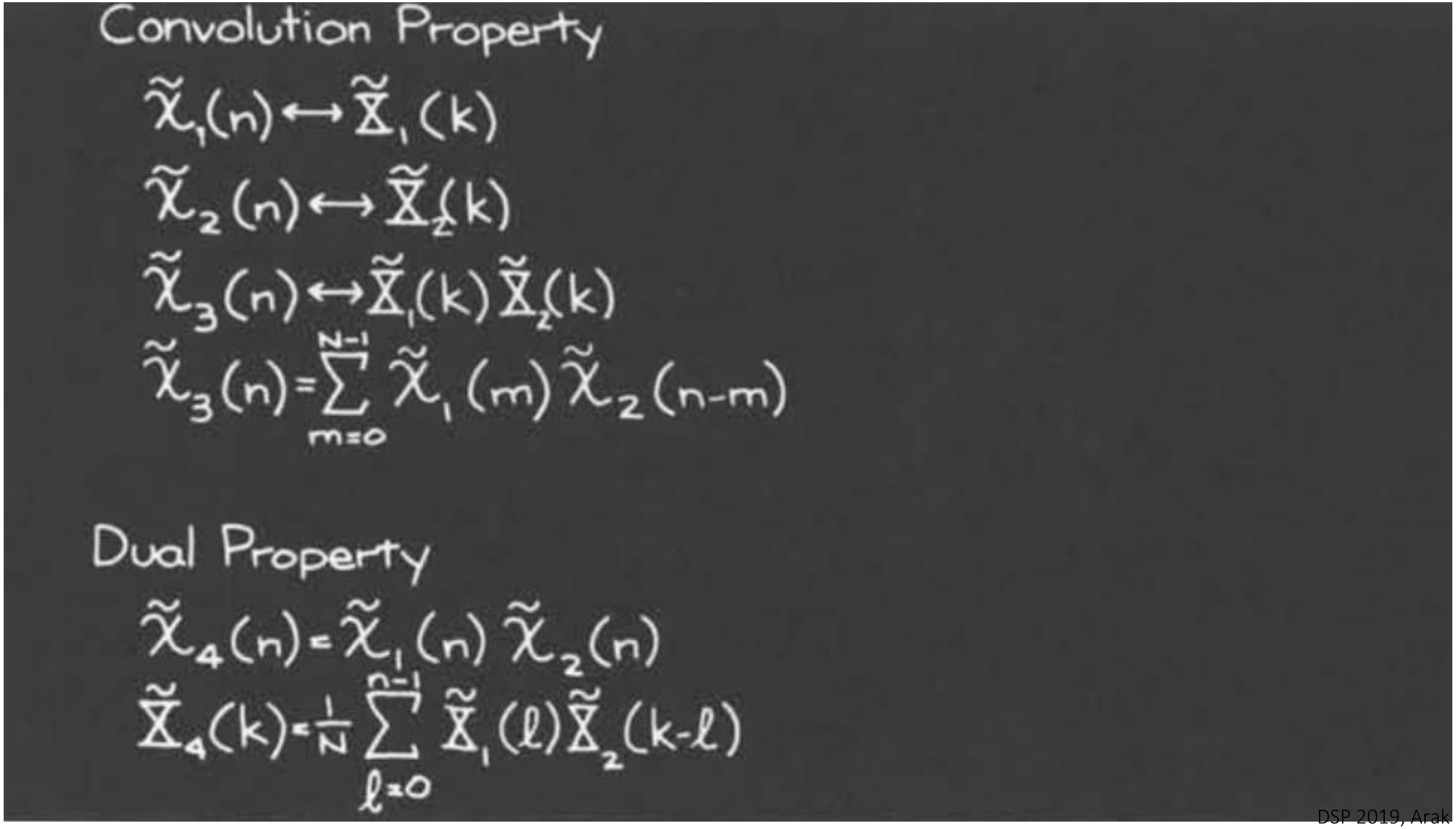
$\tilde{X}_I(k) = -\tilde{X}_I(-k)$ odd

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

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

$\measuredangle\tilde{X}(k)$ odd

Class Review



Convolution Property

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

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

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

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

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

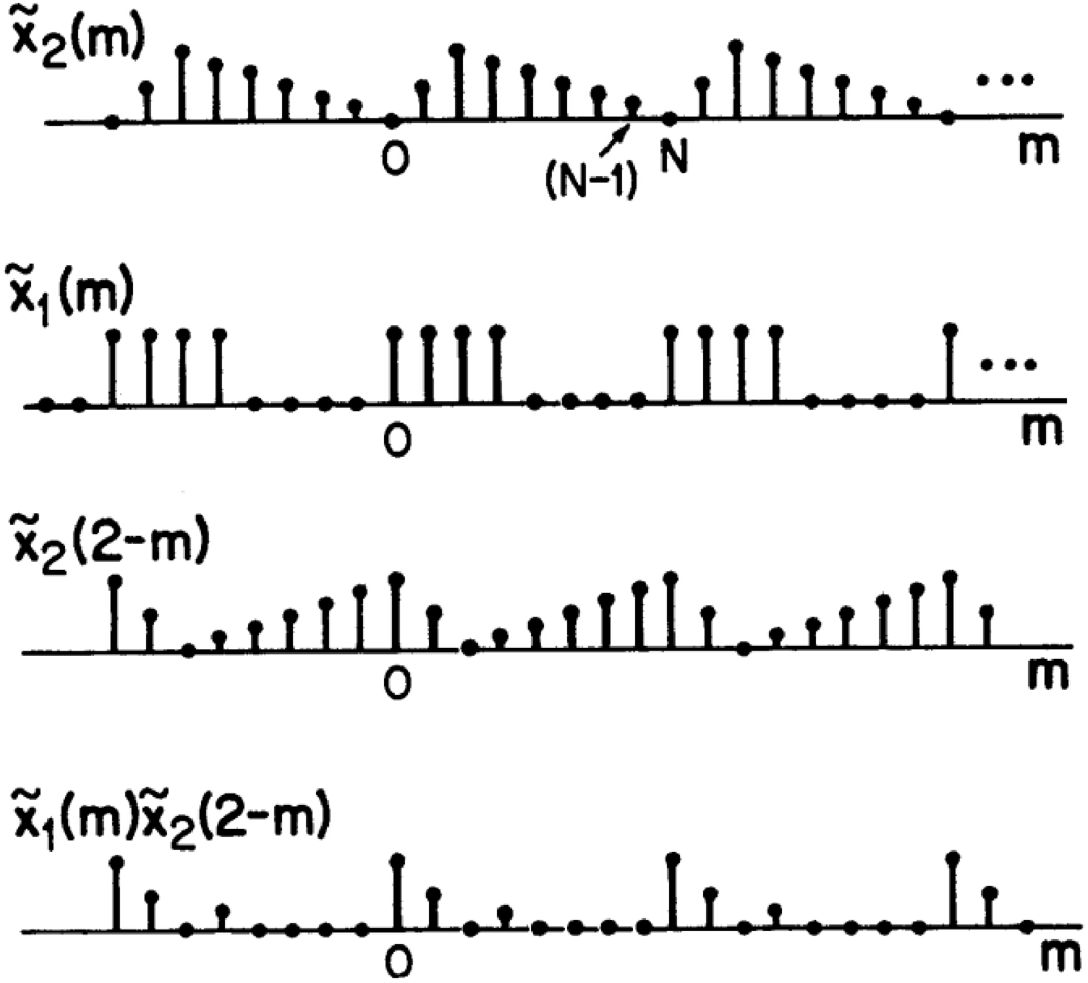
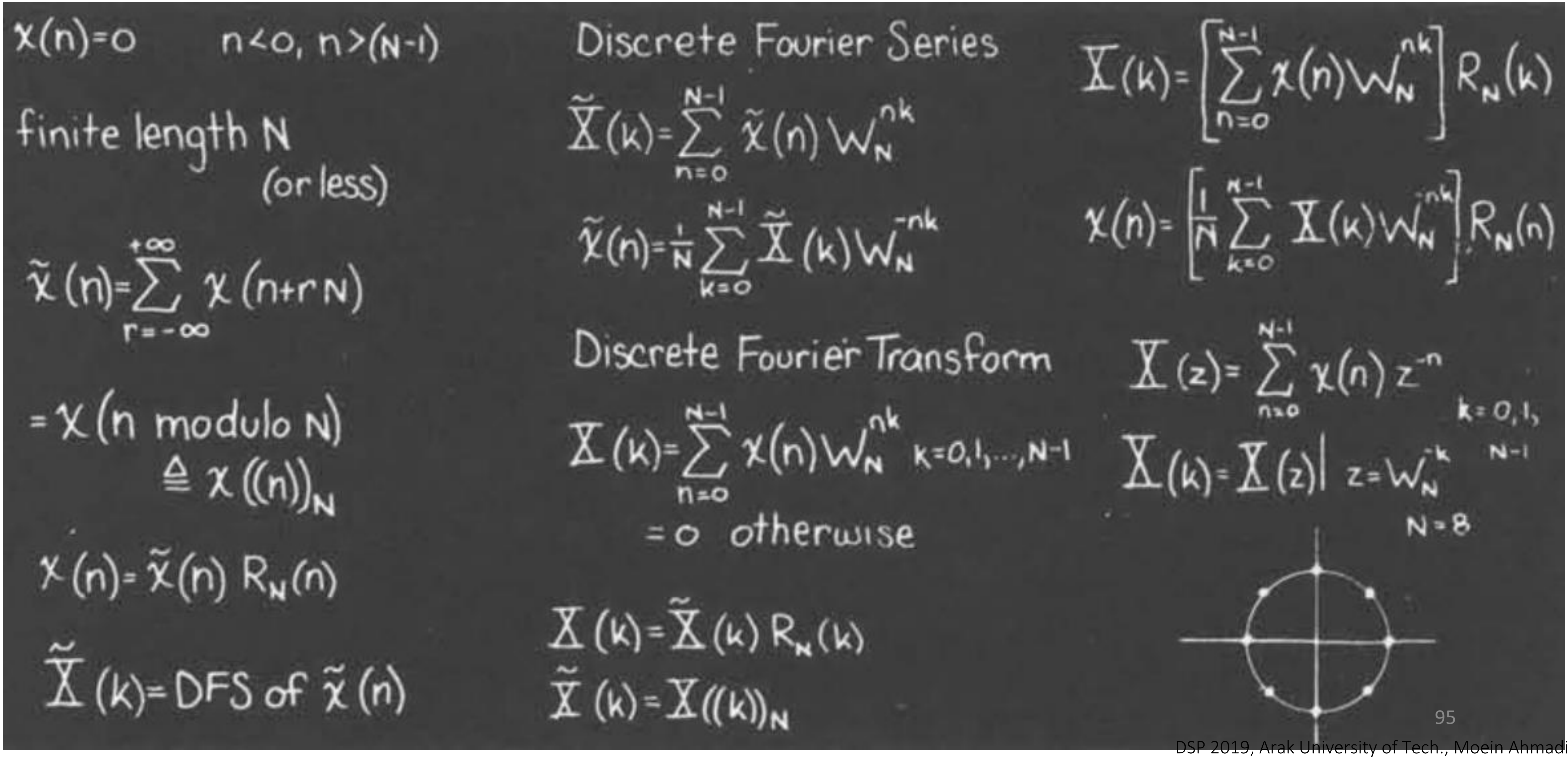


Illustration of the sequences involved in forming a periodic convolution.

Class Review



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

finite length N
(or less)

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

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

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

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

Discrete Fourier Series

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

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

Discrete Fourier Transform

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

$$=0 \quad \text{otherwise}$$

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

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

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

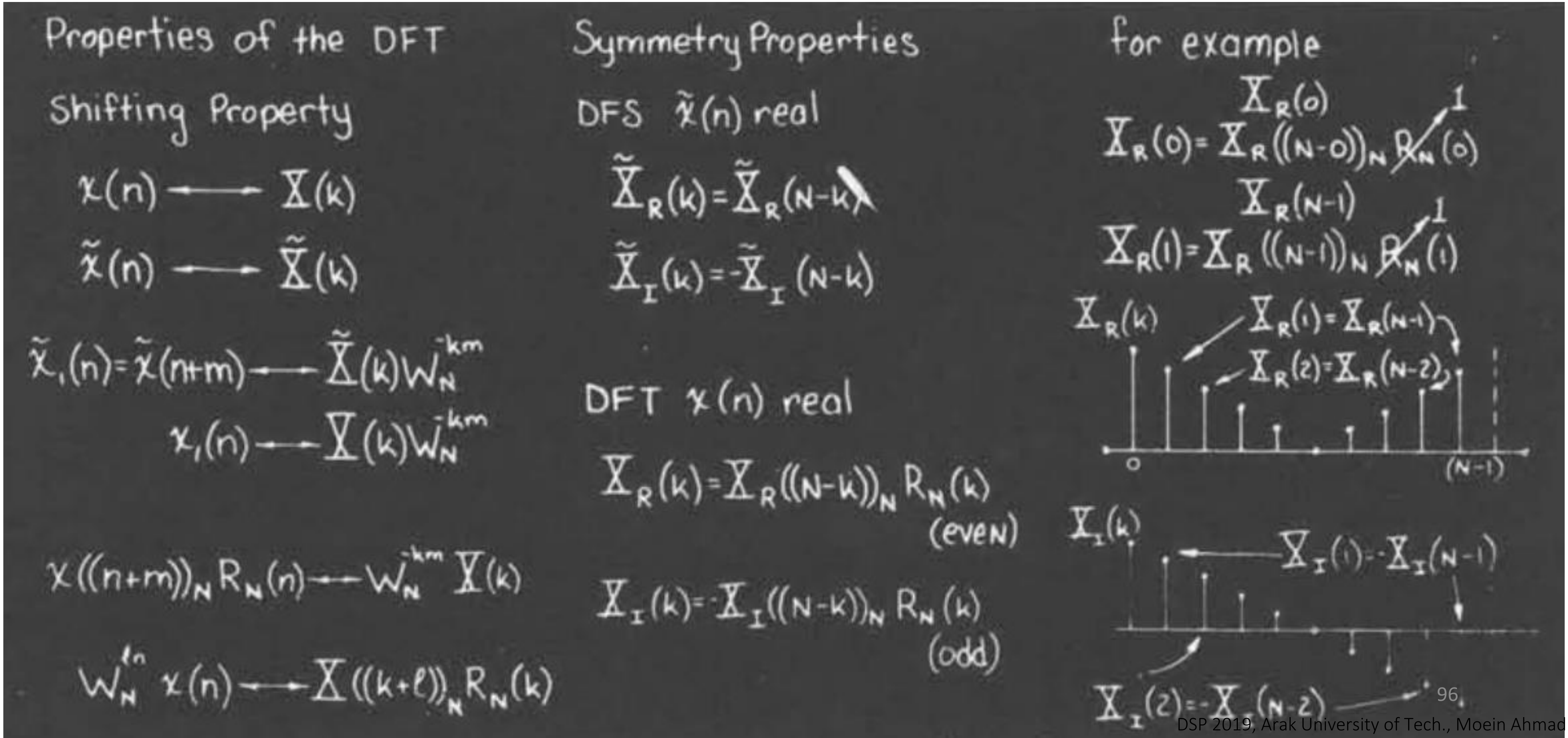
$$x(n)=\left[\frac{1}{N}\sum_{k=0}^{N-1}X(k)\,W_N^{-nk}\right]R_N(n)$$

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

$$X(k)=X(z)\Big|_{z=W_N^{-k}} \quad k=0,1,\dots,N-1$$

N=8

Class Review



Properties of the DFT

Shifting Property

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

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

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

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

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

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

Symmetry Properties

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

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

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

DFT $x(n)$ real

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

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

for example

$$X_R(0) = X_R((N-0))_N \overset{1}{\cancel{R_N(0)}} \quad \underbrace{}_{} $$

$X_R(0)$ above $X_R((N-0))_N$: $X_R(0) = \overset{X_R(0)}{X_R((N-0))_N}\, \overset{1}{\cancel{R_N}}(0)$

$X_R(1) = \overset{X_R(N-1)}{X_R((N-1))_N}\, \overset{1}{\cancel{R_N}}(1)$

$X_R(k)$

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

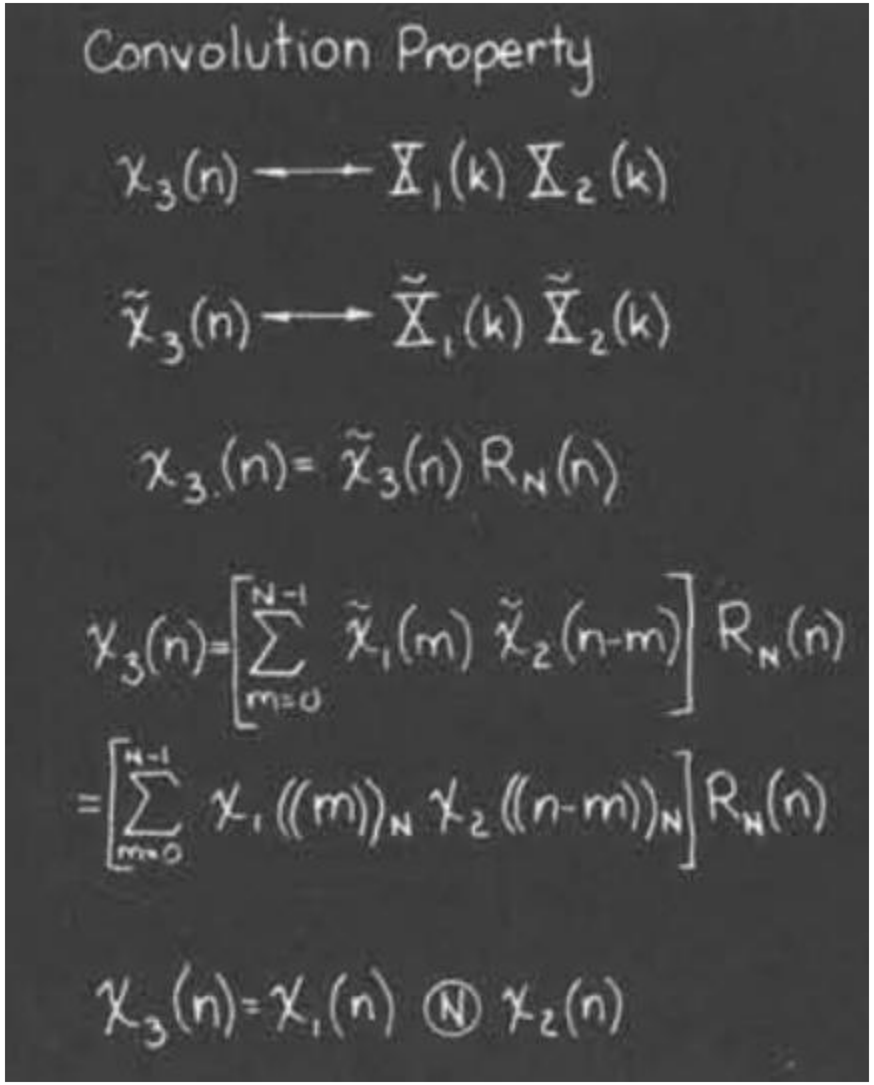
$X_R(2) = X_R(N-2)$

$X_I(k)$

$X_I(1) = -X_I(N-1)$

$X_I(2) = -X_I(N-2)$

Class Review



Convolution Property

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

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

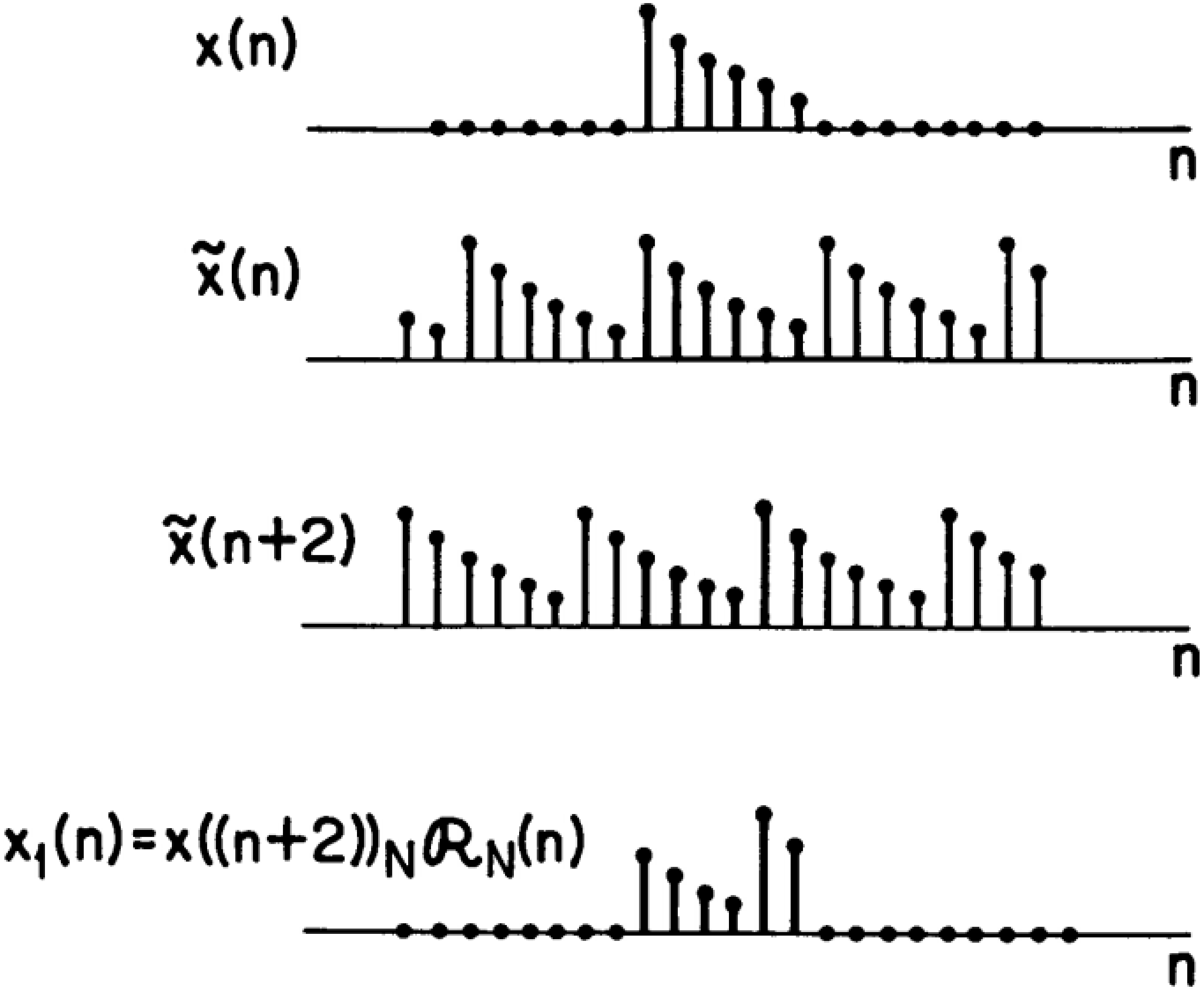
$$x_3(n) = \tilde{x}_3(n)\,R_N(n)$$

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

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

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

Class Review



Circular shifting of a finite length sequence.

Class Review

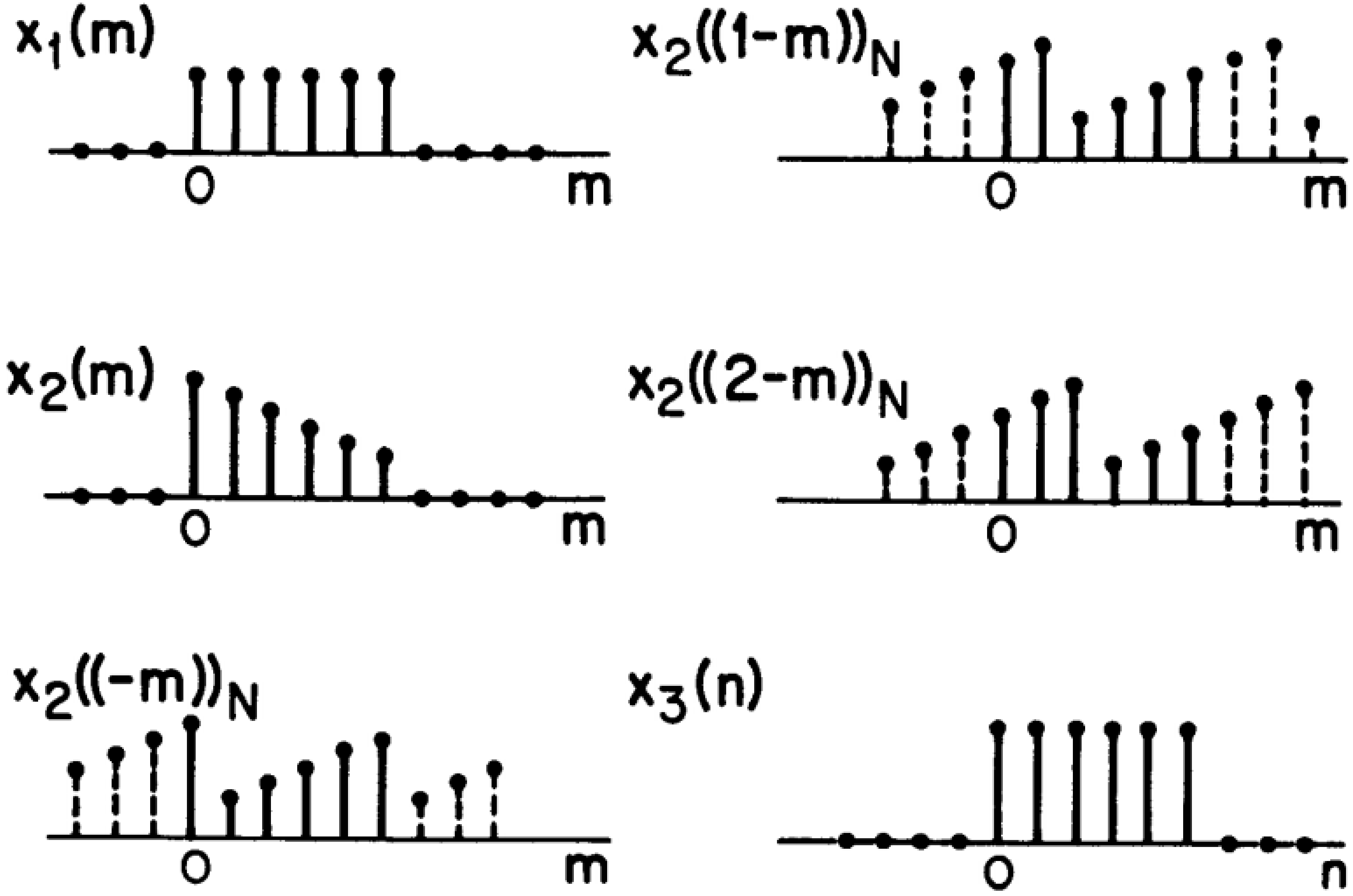
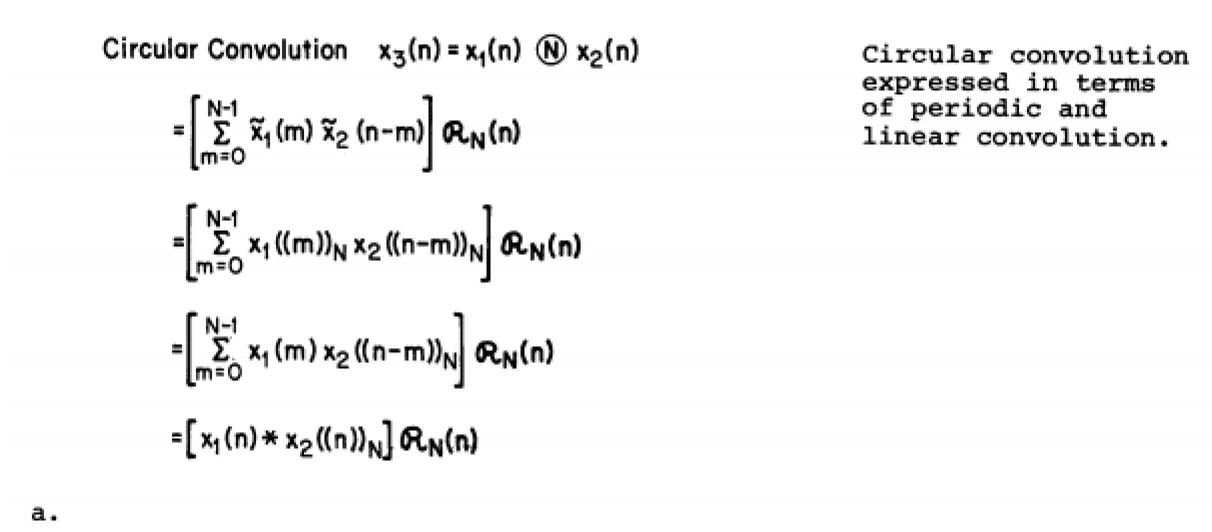


Illustration of circular convolution. (Note that $x_2((-m))_N$ is incorrectly drawn. In Problem 9.4 you are asked to correct this.)

Class Review



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

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

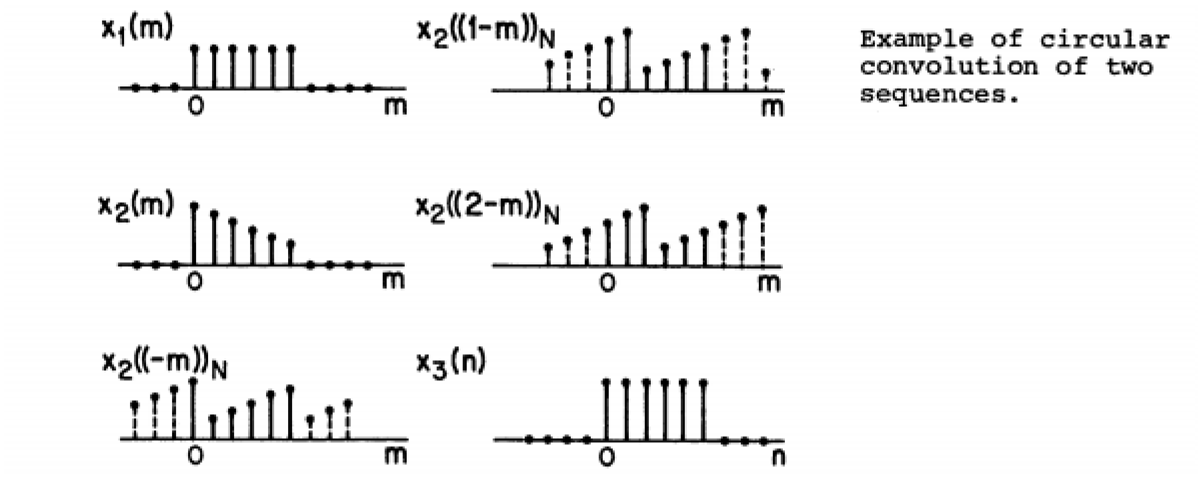
$$= \left[\sum_{m=0}^{N-1} x_1((m))_N\, x_2((n-m))_N\right] \mathcal{R}_N(n)$$

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

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

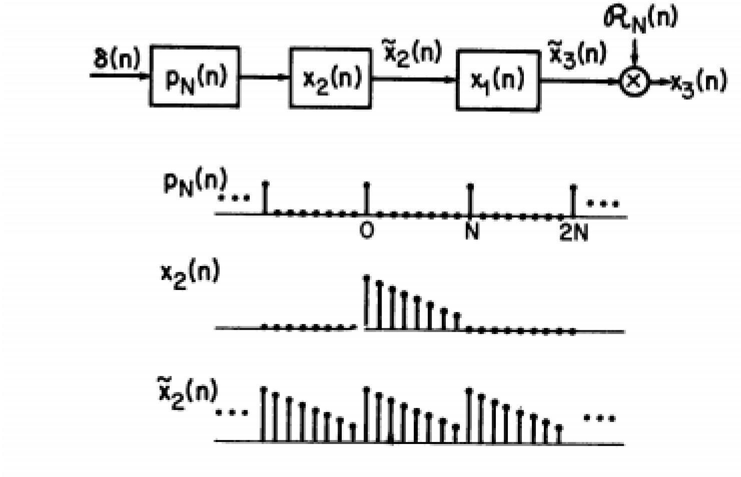
a.

Circular convolution expressed in terms of periodic and linear convolution.



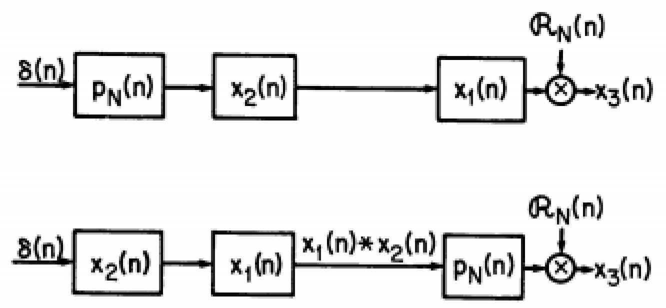
b.

Example of circular convolution of two sequences.



c.

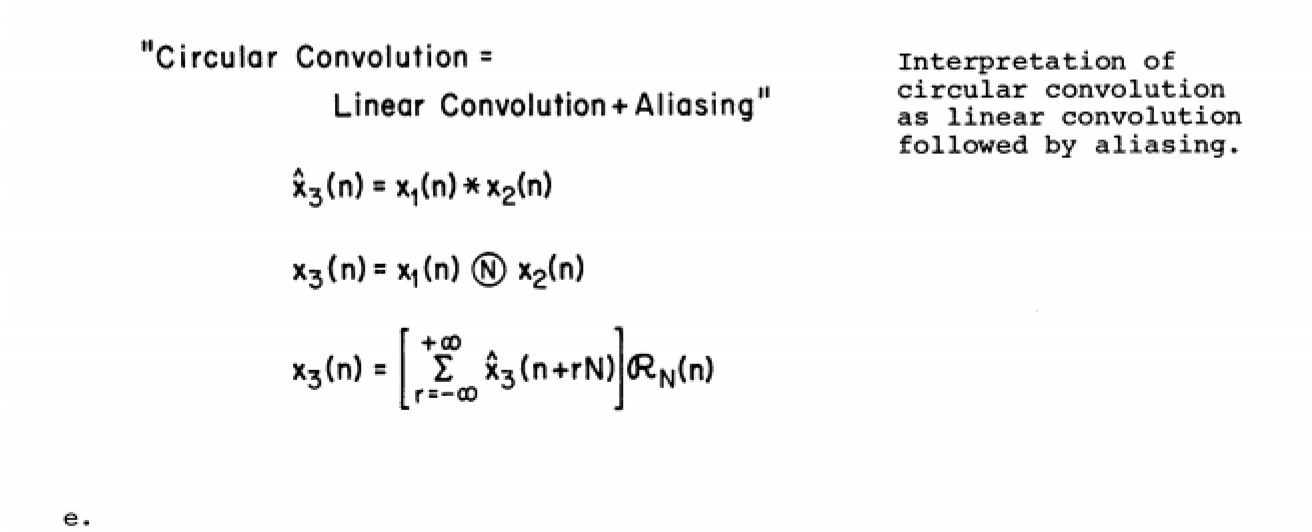
An interpretation of circular convolution.



Rearrangement of the operations in forming the circular convolution.

d.

Class Review



"Circular Convolution = Linear Convolution + Aliasing"

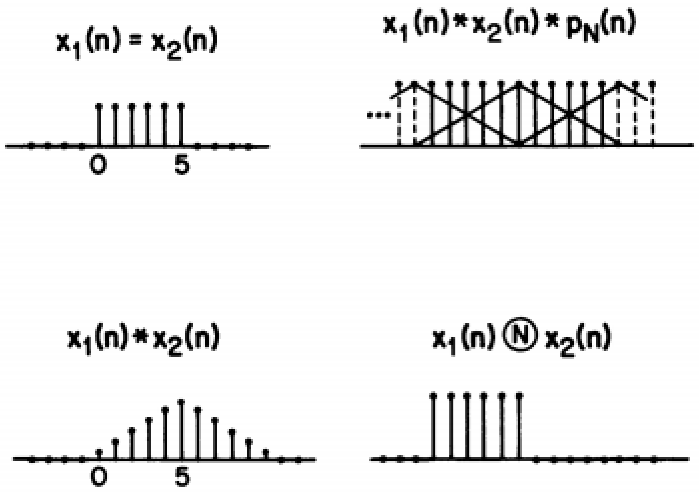
$$\hat{x}_3(n) = x_1(n) * x_2(n)$$

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

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

Interpretation of circular convolution as linear convolution followed by aliasing.

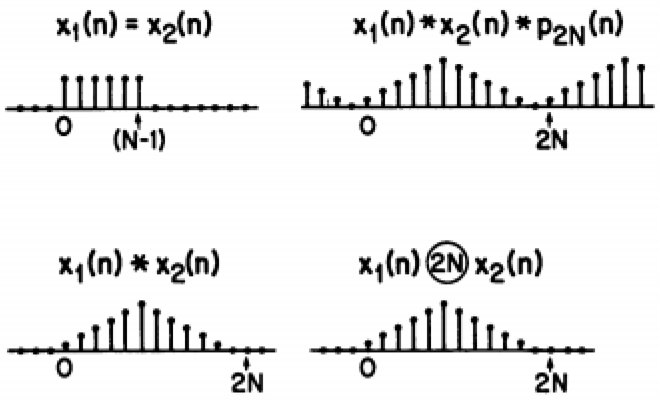
e.



$x_1(n) = x_2(n)$
0
5
$x_1(n) * x_2(n) * p_N(n)$
...
$x_1(n) * x_2(n)$
0
5
$x_1(n) \; \textcircled{N} \; x_2(n)$

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

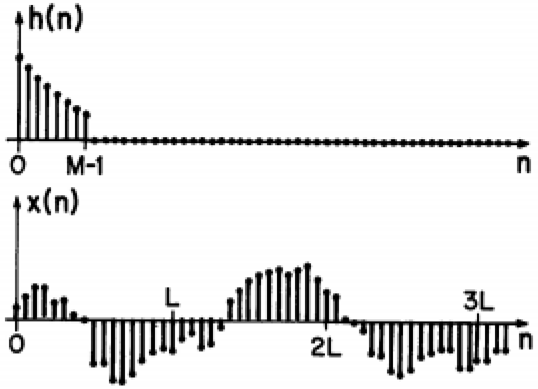
f.



$x_1(n) = x_2(n)$
0
(N-1)
$x_1(n) * x_2(n) * p_{2N}(n)$
0
2N
$x_1(n) * x_2(n)$
0
2N
$x_1(n) \; \textcircled{2N} \; x_2(n)$
0
2N

Obtaining a linear convolution through the use of circular convolution.

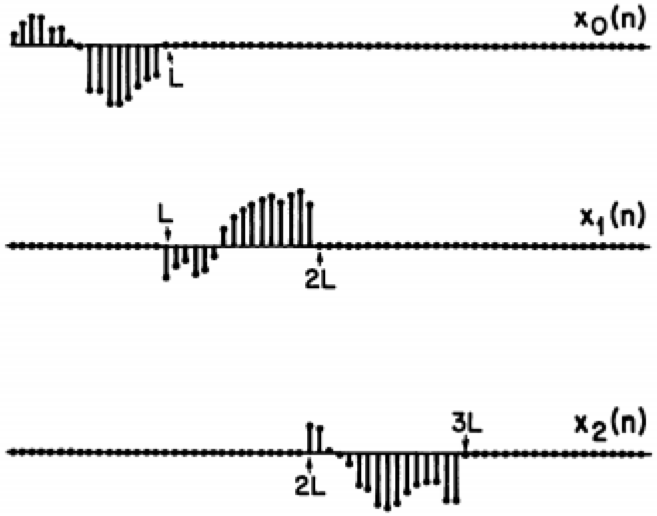
g.



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

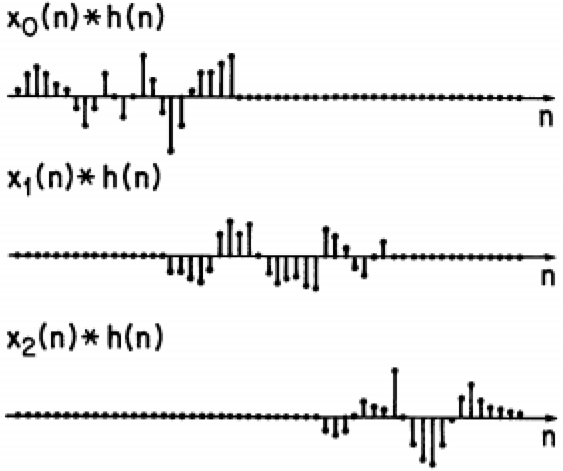
h.

Class Review



i.

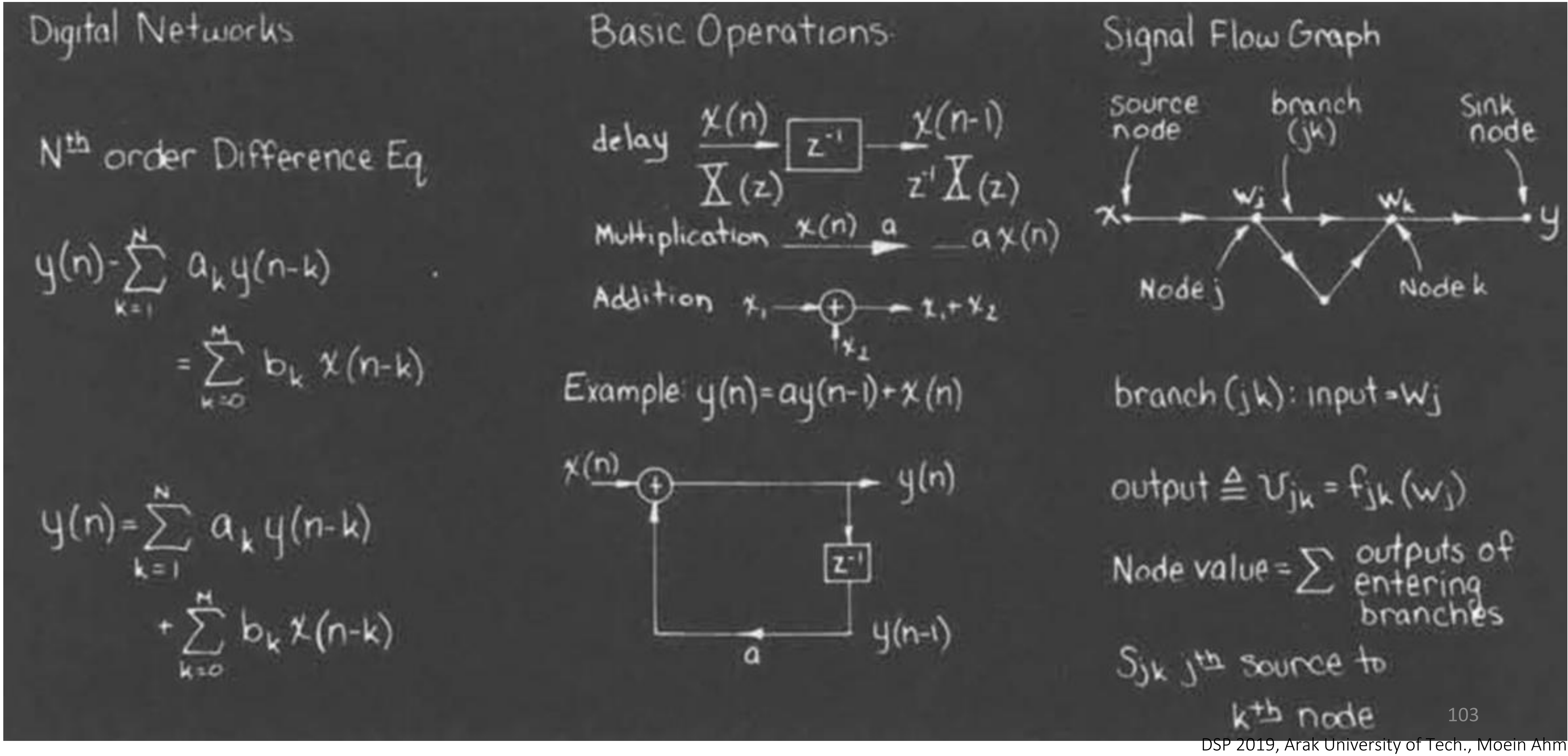
Sectioning of the sequence x(n).



j.

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

Class Review



Digital Networks

N^{th} order Difference Eq

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

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

Basic Operations:

delay $\xrightarrow{x(n)} \boxed{z^{-1}} \xrightarrow{x(n-1)}$

$X(z)$ $\quad$ $z^{-1}X(z)$

Multiplication $\xrightarrow[x(n)]{a}$ $a\,x(n)$

Addition $x_1 \to \oplus \to x_1 + x_2$, $\uparrow x_2$

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

$x(n)$
$y(n)$
z^{-1}
$y(n-1)$
a

Signal Flow Graph

source node
branch (jk)
sink node
x
w_j
w_k
y
Node j
Node k

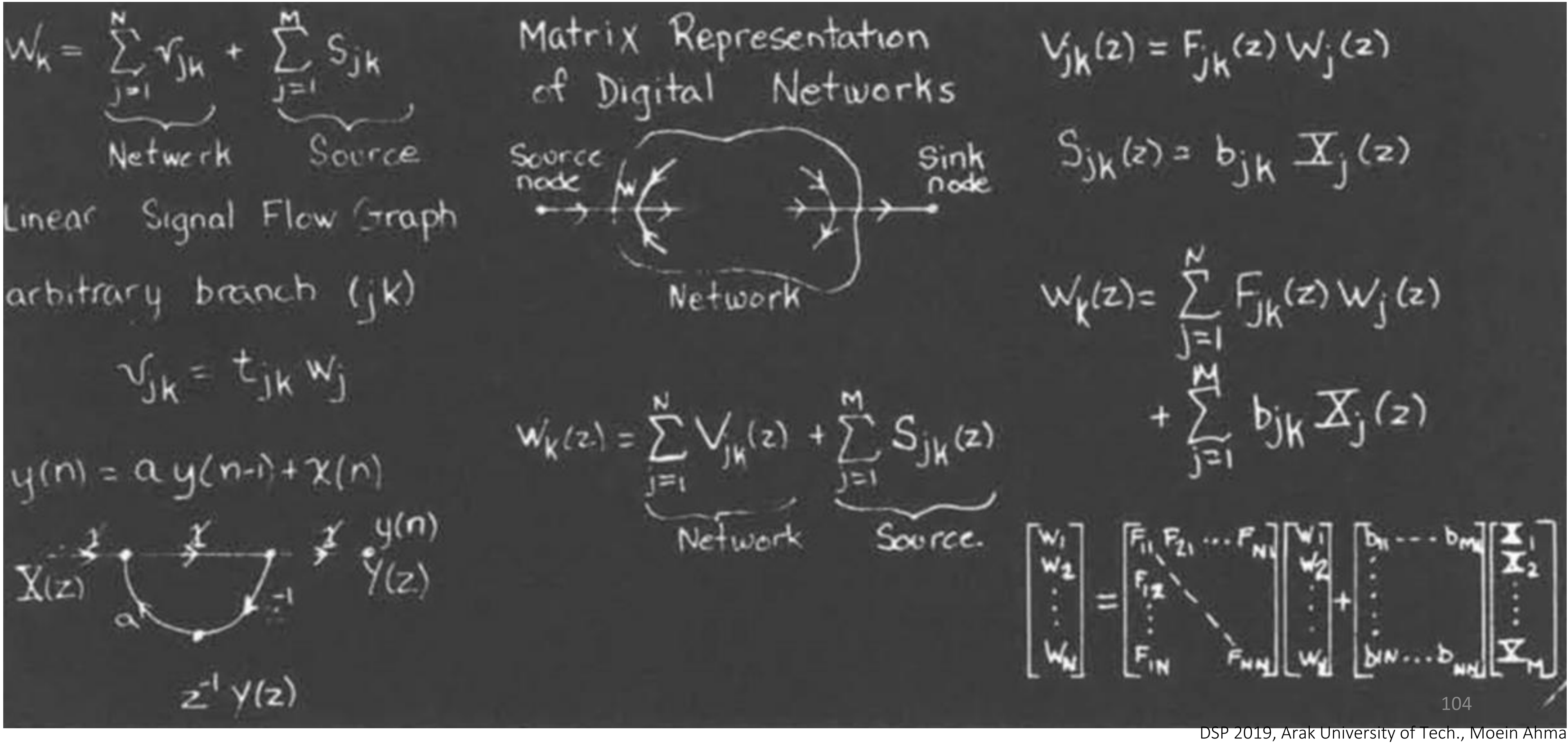
branch (jk): input $= w_j$

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

Node value $= \sum$ outputs of entering branches

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

Class Review



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

Linear Signal Flow Graph

arbitrary branch (jk)

$$v_{jk} = t_{jk} w_j$$

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

Matrix Representation of Digital Networks

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

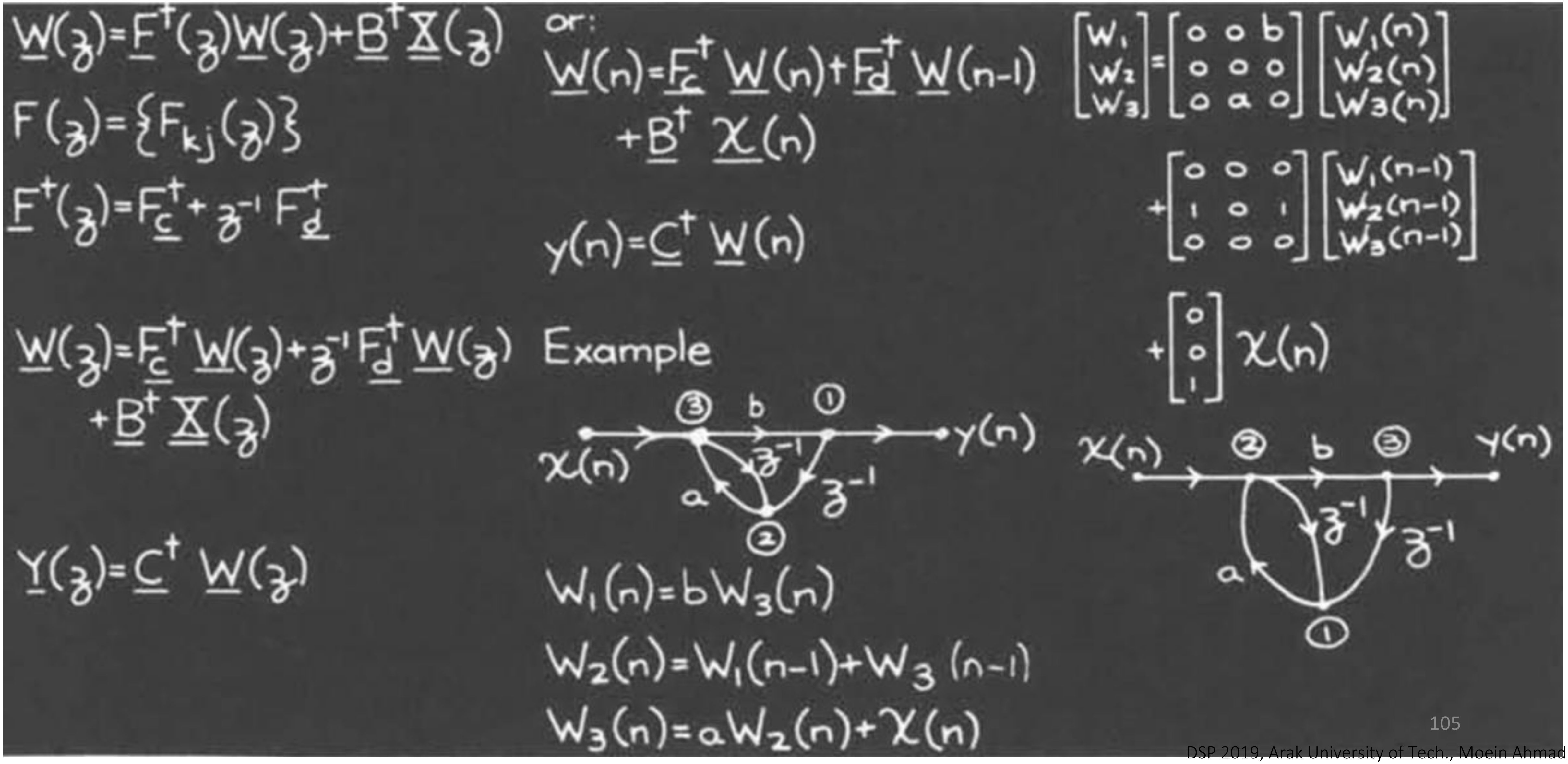
$$V_{jk}(z) = F_{jk}(z) W_j(z)$$

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

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

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

Class Review



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

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

$$\underline{F}^t(z) = \underline{F}_c^t + z^{-1}\underline{F}_d^t$$

$$\underline{W}(z) = \underline{F}_c^t\underline{W}(z) + z^{-1}\underline{F}_d^t\underline{W}(z) + \underline{B}^t\underline{X}(z)$$

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

or:

$$\underline{W}(n) = \underline{F}_c^t\underline{W}(n) + \underline{F}_d^t\underline{W}(n-1) + \underline{B}^t\underline{x}(n)$$

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

Example

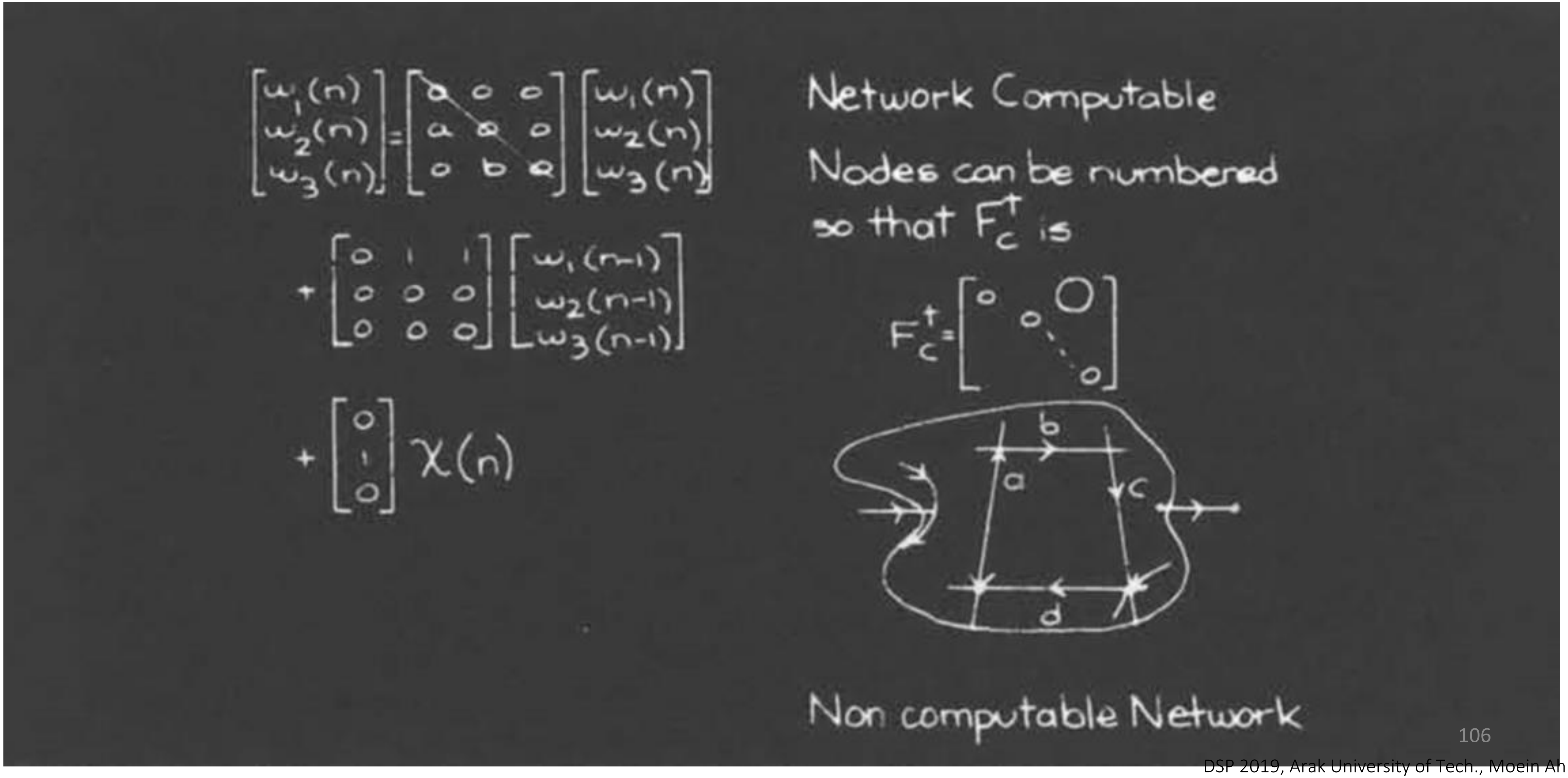
$$W_1(n) = bW_3(n)$$

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

$$W_3(n) = aW_2(n) + x(n)$$

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

Class Review



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

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

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

Network Computable

Nodes can be numbered so that F_c^t is

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

Non computable Network

Class Review

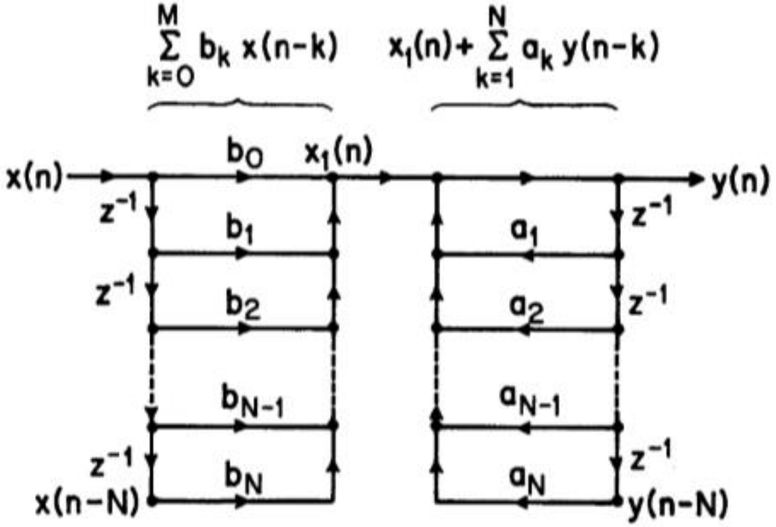
$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1-\sum_{k=1}^{N} a_k z^{-k}}$$

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

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

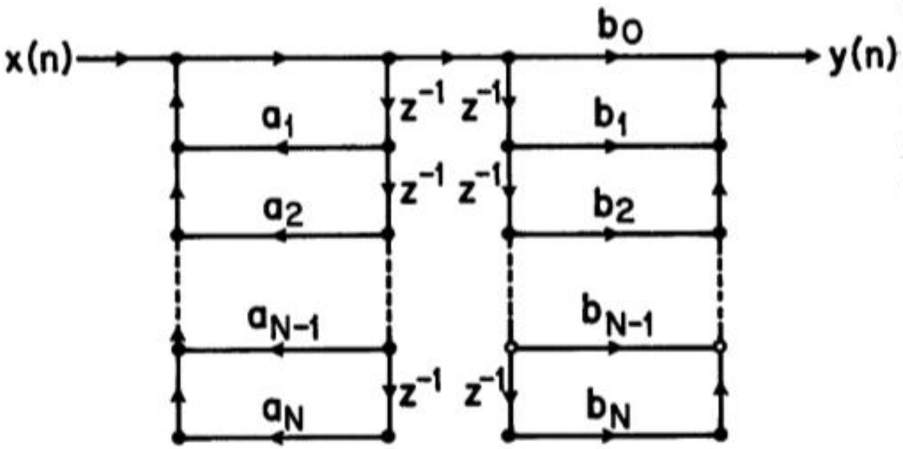
z-transform and difference equation for a general IIR system.

a.



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

Class Review



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

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

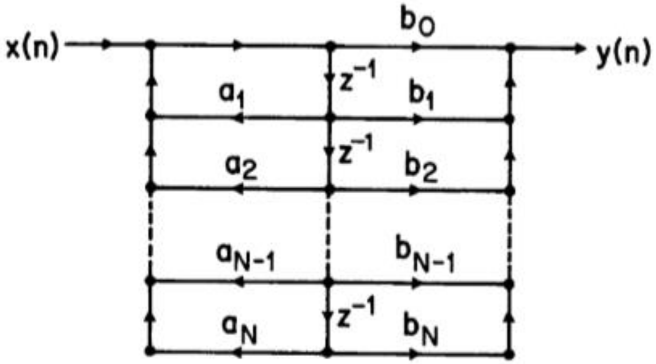
$$H(z) = \left[\frac{1}{1-\sum_{k=1}^{N} a_k z^{-k}}\right]\left[\sum_{k=0}^{M} b_k z^{-k}\right]$$

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

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

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

Class Review



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

TRANSPOSITION THEOREM

1. REVERSE DIRECTION OF ALL BRANCHES

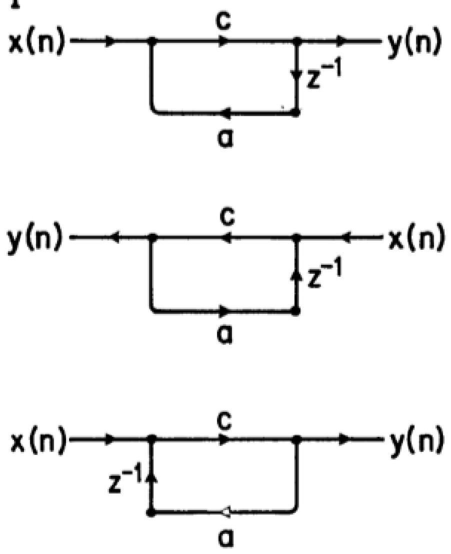
2. INTERCHANGE INPUT AND OUTPUT*

TRANSFER FUNCTION REMAINS THE SAME

Transposition theorem for signal flow-graphs.

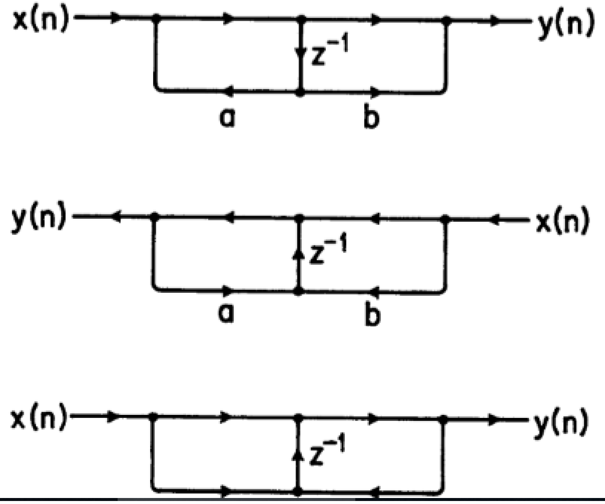
Class Review

Example 1



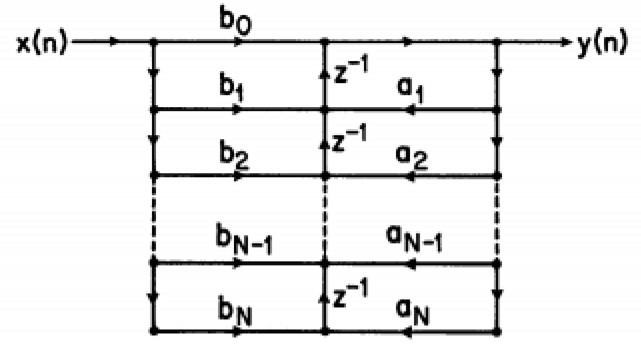
Example of Transposition theorem.

Example 2



Example of Transposition theorem.

Class Review



Transposed direct form II structure.

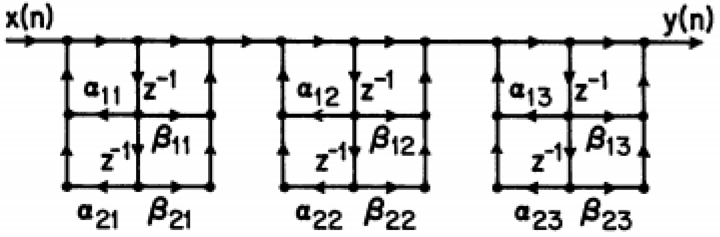
Cascade Structure

Factorization of the z-transform for the cascade structure.

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

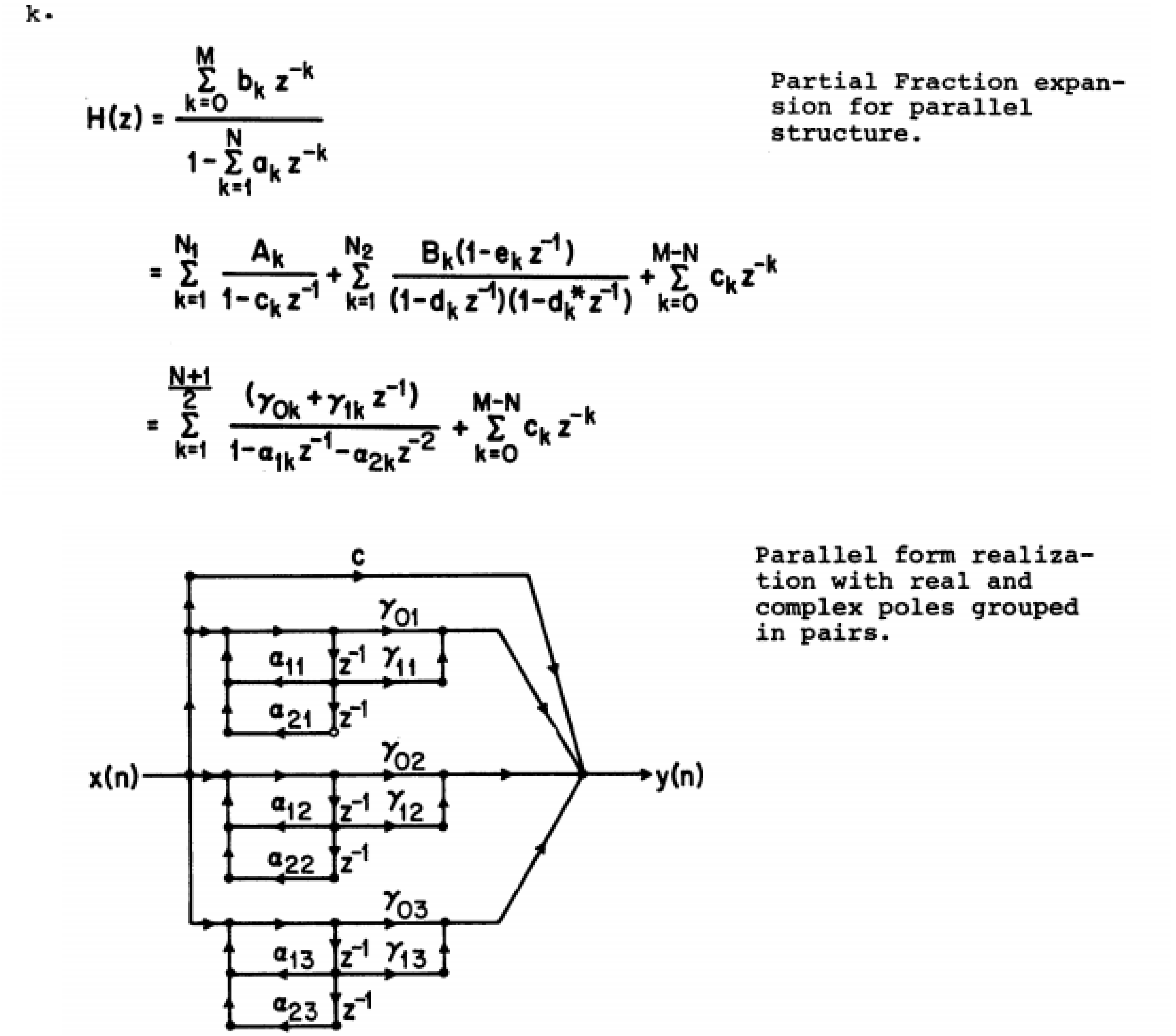
$$= A\Pi \frac{1+\beta_{1k} z^{-1}+\beta_{2k} z^{-2}}{1-\alpha_{1k} z^{-1}-\alpha_{2k} z^{-2}}$$

j.



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

Class Review



k.

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

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

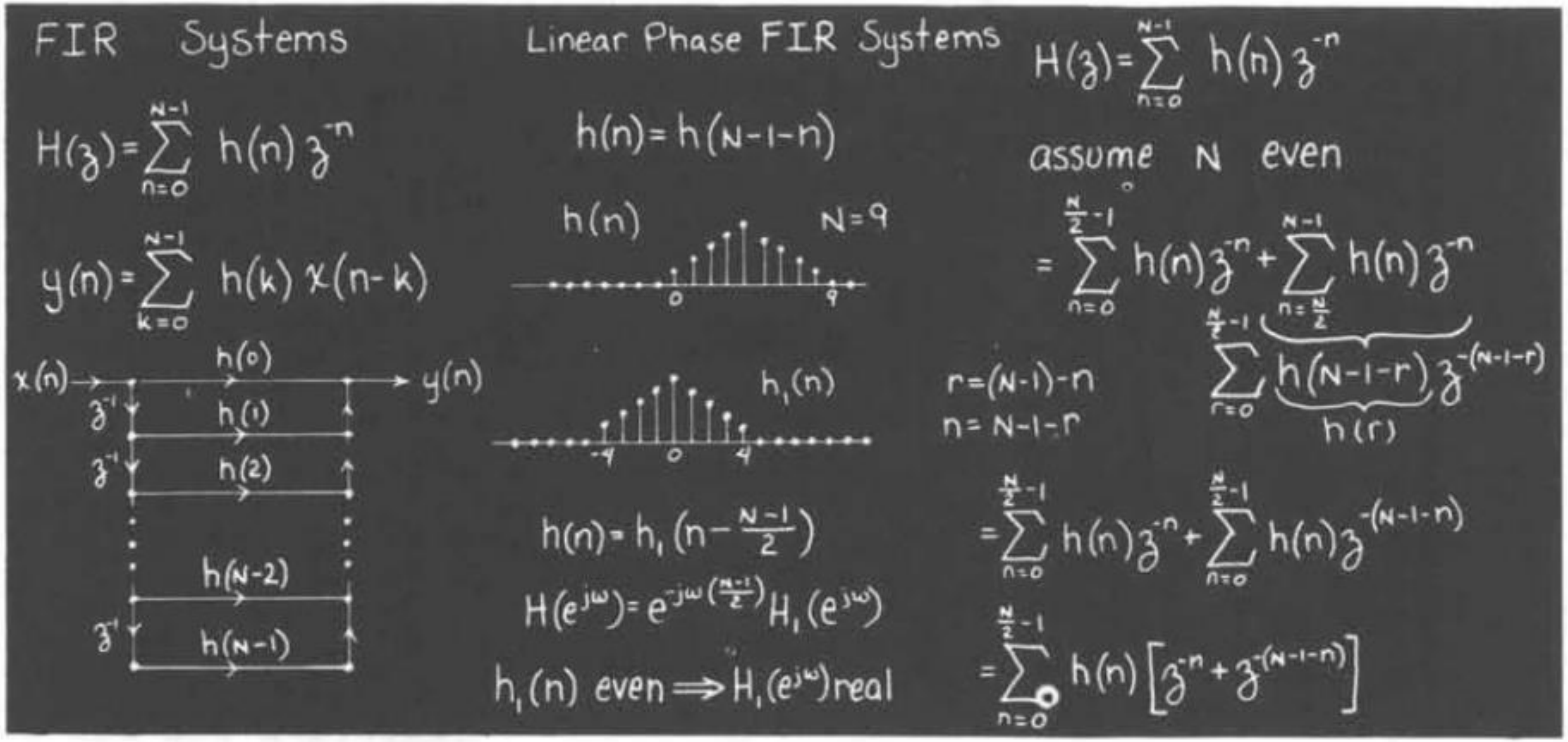
$$= \sum_{k=1}^{\frac{N+1}{2}} \frac{(\gamma_{0k}+\gamma_{1k} z^{-1})}{1-\alpha_{1k} z^{-1}-\alpha_{2k} z^{-2}} + \sum_{k=0}^{M-N} c_k z^{-k}$$

Partial Fraction expansion for parallel structure.

c
γ_{01}
α_{11}
z^{-1}
γ_{11}
α_{21}
z^{-1}
γ_{02}
x(n)
y(n)
α_{12}
z^{-1}
γ_{12}
α_{22}
z^{-1}
γ_{03}
α_{13}
z^{-1}
γ_{13}
α_{23}
z^{-1}

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

Class Review



FIR Systems

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

$$y(n)=\sum_{k=0}^{N-1} h(k)\, x(n-k)$$

x(n)
h(0)
y(n)
z^{-1}
h(1)
z^{-1}
h(2)
h(N-2)
z^{-1}
h(N-1)

Linear Phase FIR Systems

$$h(n)=h(N-1-n)$$

h(n)
N=9
0
9
$h_1(n)$
-4
0
4

$$h(n)=h_1\left(n-\frac{N-1}{2}\right)$$

$$H(e^{j\omega})=e^{-j\omega\left(\frac{N-1}{2}\right)}H_1(e^{j\omega})$$

$h_1(n)$ even $\Longrightarrow H_1(e^{j\omega})$ real

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

assume N even

$$=\sum_{n=0}^{\frac{N}{2}-1} h(n) z^{-n}+\overbrace{\sum_{n=\frac{N}{2}}^{N-1} h(n) z^{-n}}^{\sum_{r=0}^{\frac{N}{2}-1}\underbrace{h(N-1-r)}_{h(r)} z^{-(N-1-r)}}$$

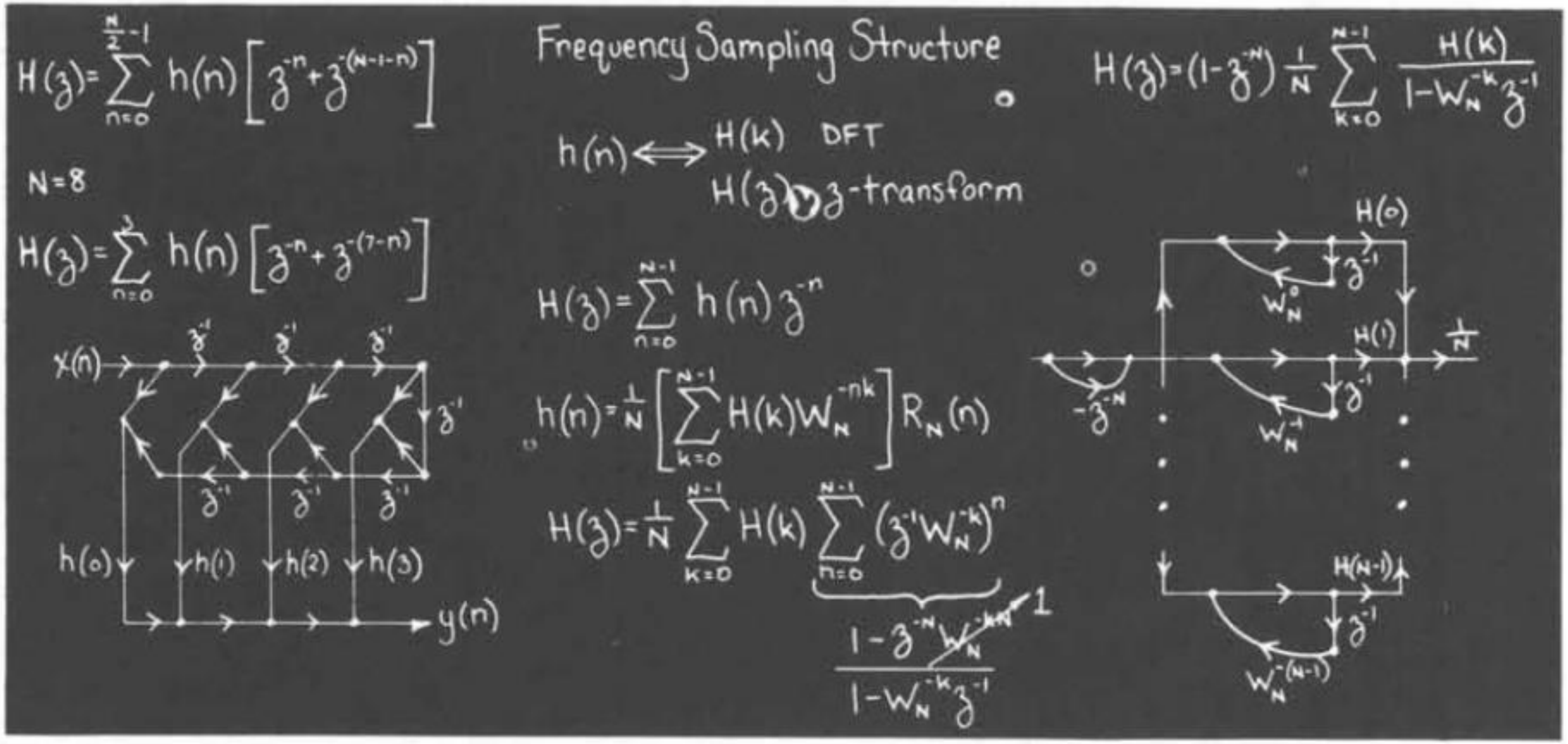
$r=(N-1)-n$

$n=N-1-r$

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

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

Class Review



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

N=8

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

Frequency Sampling Structure

$h(n) \Longleftrightarrow H(k)$ DFT

$H(z) \Longleftrightarrow$ z-transform

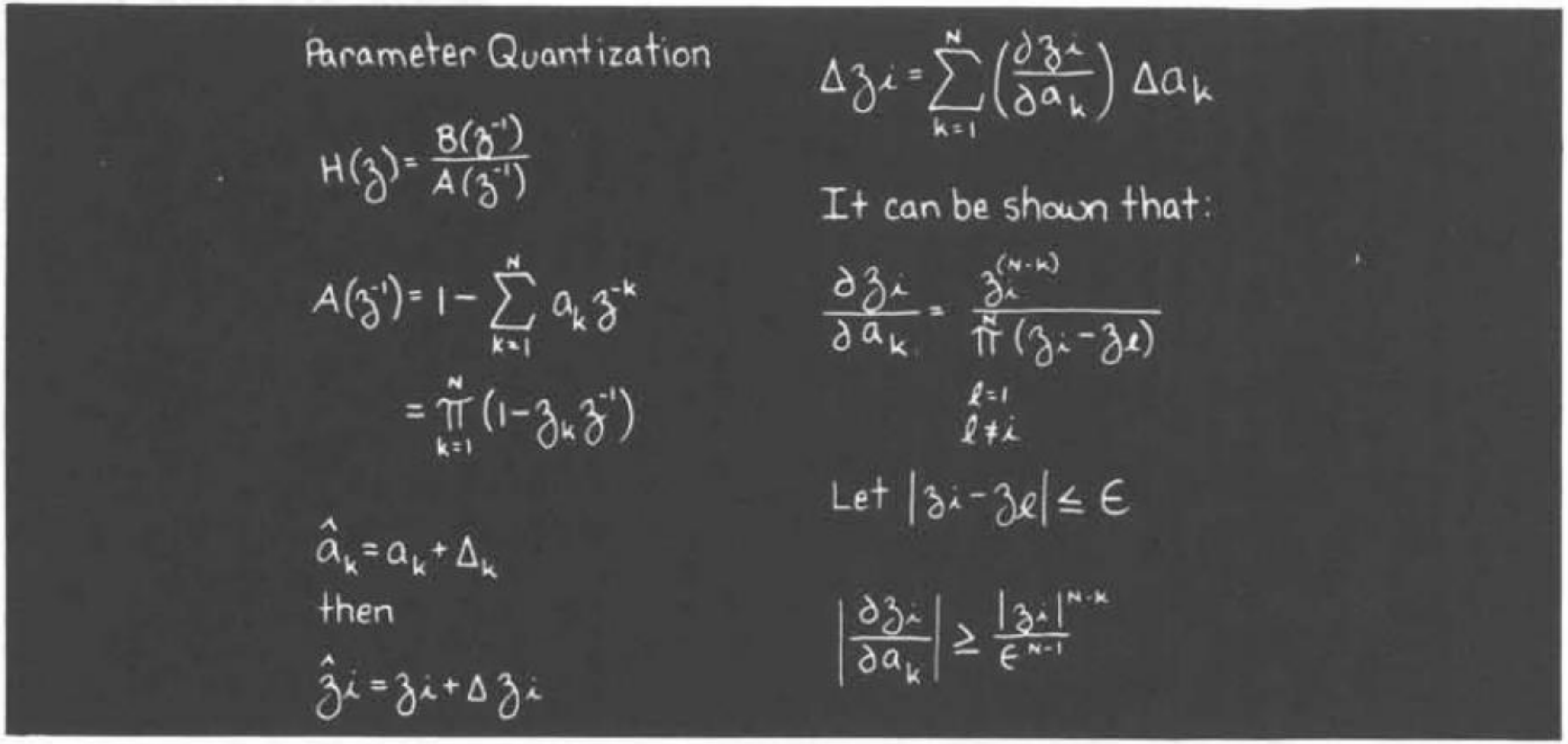
$$H(z)=\sum_{n=0}^{N-1} h(n)z^{-n}$$

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

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

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

Class Review



Parameter Quantization

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

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

$$=\prod_{k=1}^{N}\left(1-z_k z^{-1}\right)$$

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

then

$$\hat{z}_i=z_i+\Delta z_i$$

$$\Delta z_i=\sum_{k=1}^{N}\left(\frac{\partial z_i}{\partial a_k}\right)\Delta a_k$$

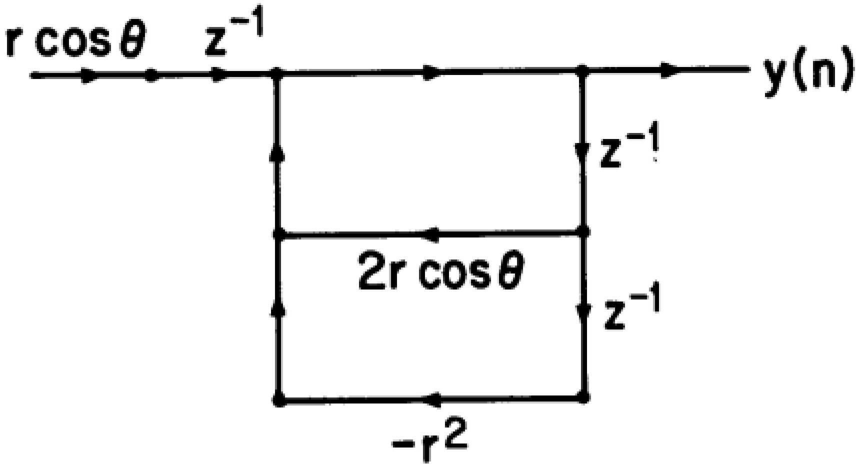
It can be shown that:

$$\frac{\partial z_i}{\partial a_k}=\frac{z_i^{(N-k)}}{\prod_{\substack{l=1\\ l\neq i}}^{N}(z_i-z_l)}$$

Let $|z_i-z_l|\leq \epsilon$

$$\left|\frac{\partial z_i}{\partial a_k}\right|\geq\frac{|z_i|^{N-k}}{\epsilon^{N-1}}$$

Class Review

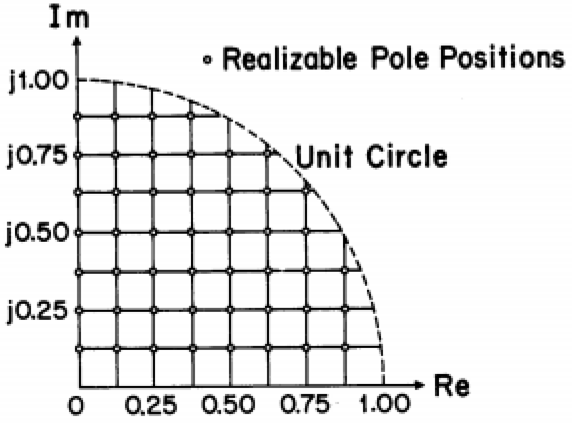


Direct-form implementation of a complex conjugate pole pair.

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

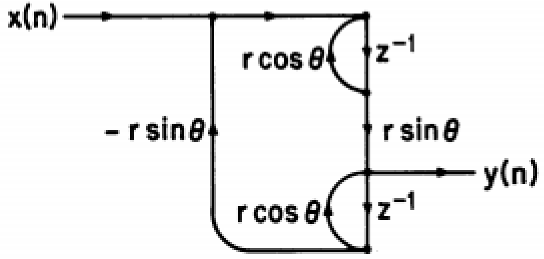
Class Review

d.



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

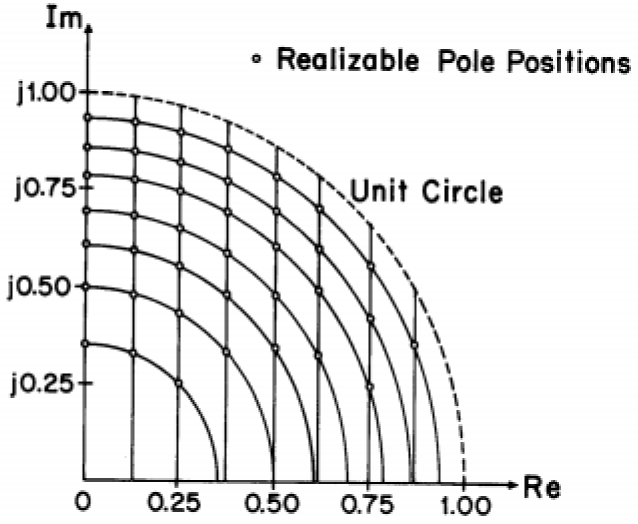
e.



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

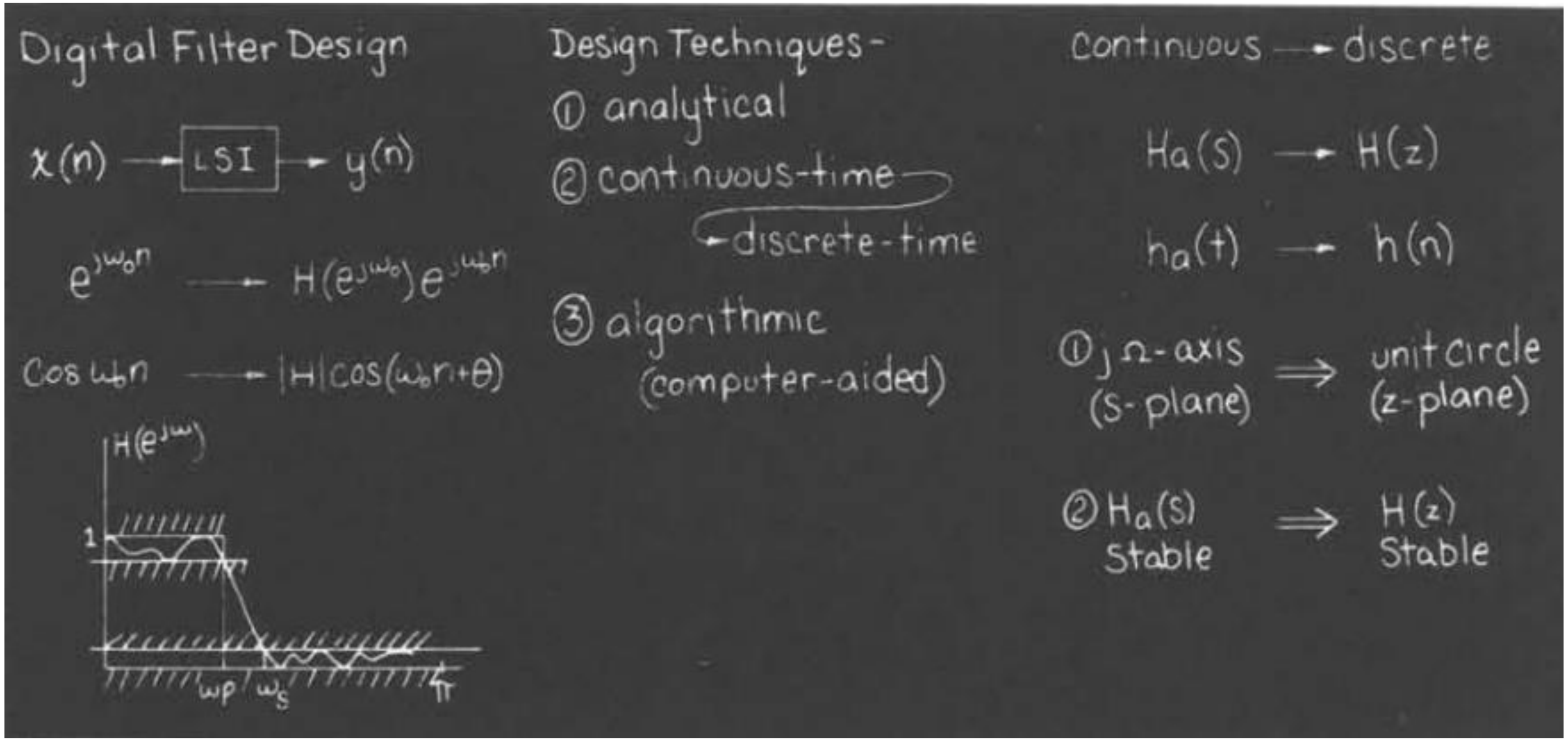
$$H(z) = \frac{r \sin \theta \, z^{-1}}{\left(1 - re^{j\theta} z^{-1}\right)\left(1 - re^{-j\theta} z^{-1}\right)}$$

Class Review



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

Class Review



Digital Filter Design

$x(n) \longrightarrow \boxed{\text{LSI}} \longrightarrow y(n)$

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

$\cos \omega_0 n \longrightarrow |H|\cos(\omega_0 n+\theta)$

$H(e^{j\omega})$
1
ω_p
ω_s
π

Design Techniques -

① analytical

② continuous-time $\rightarrow$ discrete-time

③ algorithmic
(computer-aided)

Continuous $\rightarrow$ discrete

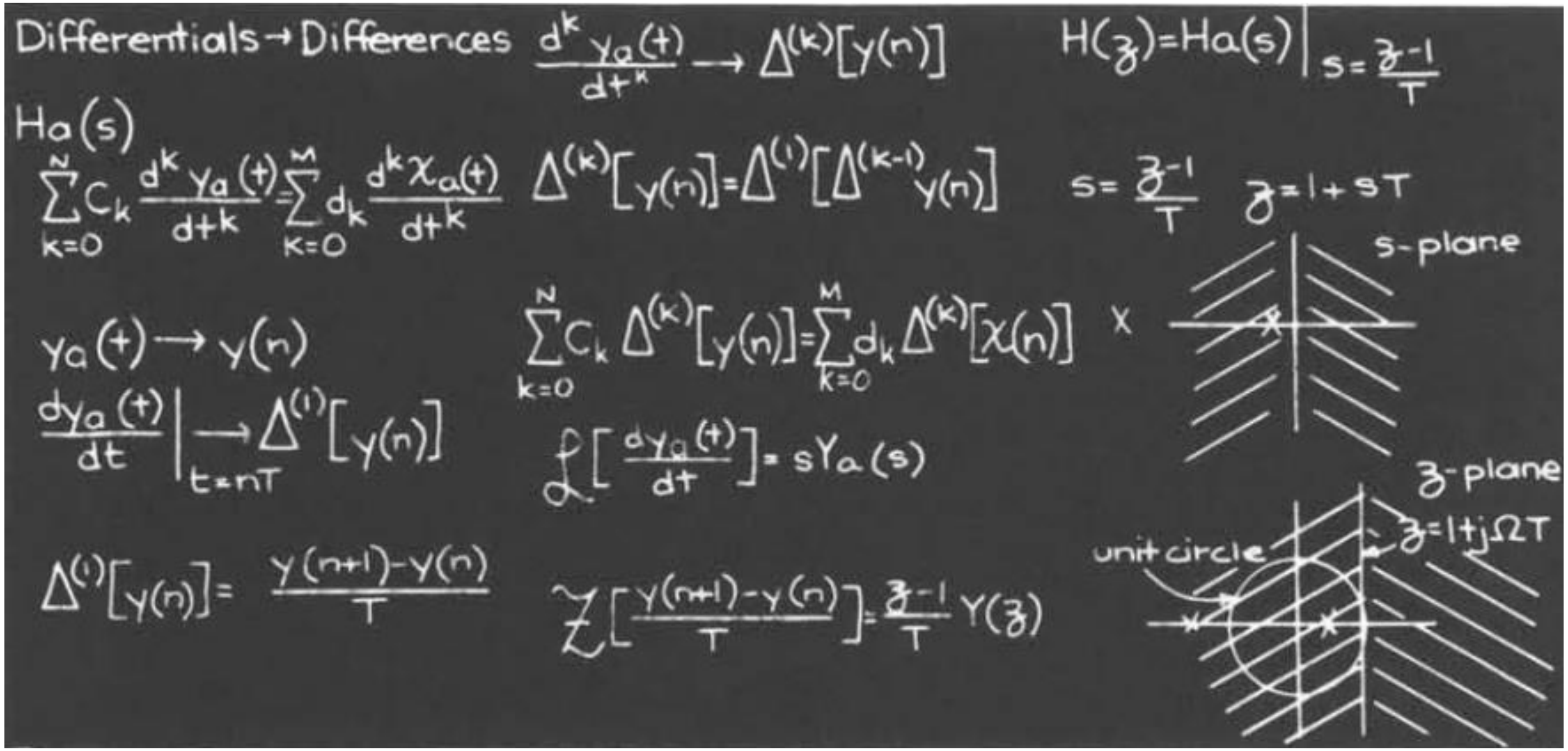
$H_a(S) \rightarrow H(z)$

$h_a(t) \rightarrow h(n)$

① $j\Omega$-axis (S-plane) $\Rightarrow$ unit circle (z-plane)

② $H_a(S)$ Stable $\Rightarrow$ $H(z)$ Stable

Class Review



Differentials → Differences $\frac{d^k y_a(t)}{dt^k} \longrightarrow \Delta^{(k)}[y(n)]$

$H(z)=H_a(s)\Big|_{s=\frac{z-1}{T}}$

$H_a(s)$

$\sum_{k=0}^{N} c_k \frac{d^k y_a(t)}{dt^k} = \sum_{k=0}^{M} d_k \frac{d^k x_a(t)}{dt^k}$

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

$s=\frac{z-1}{T}$ $z=1+sT$

$y_a(t) \longrightarrow y(n)$

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

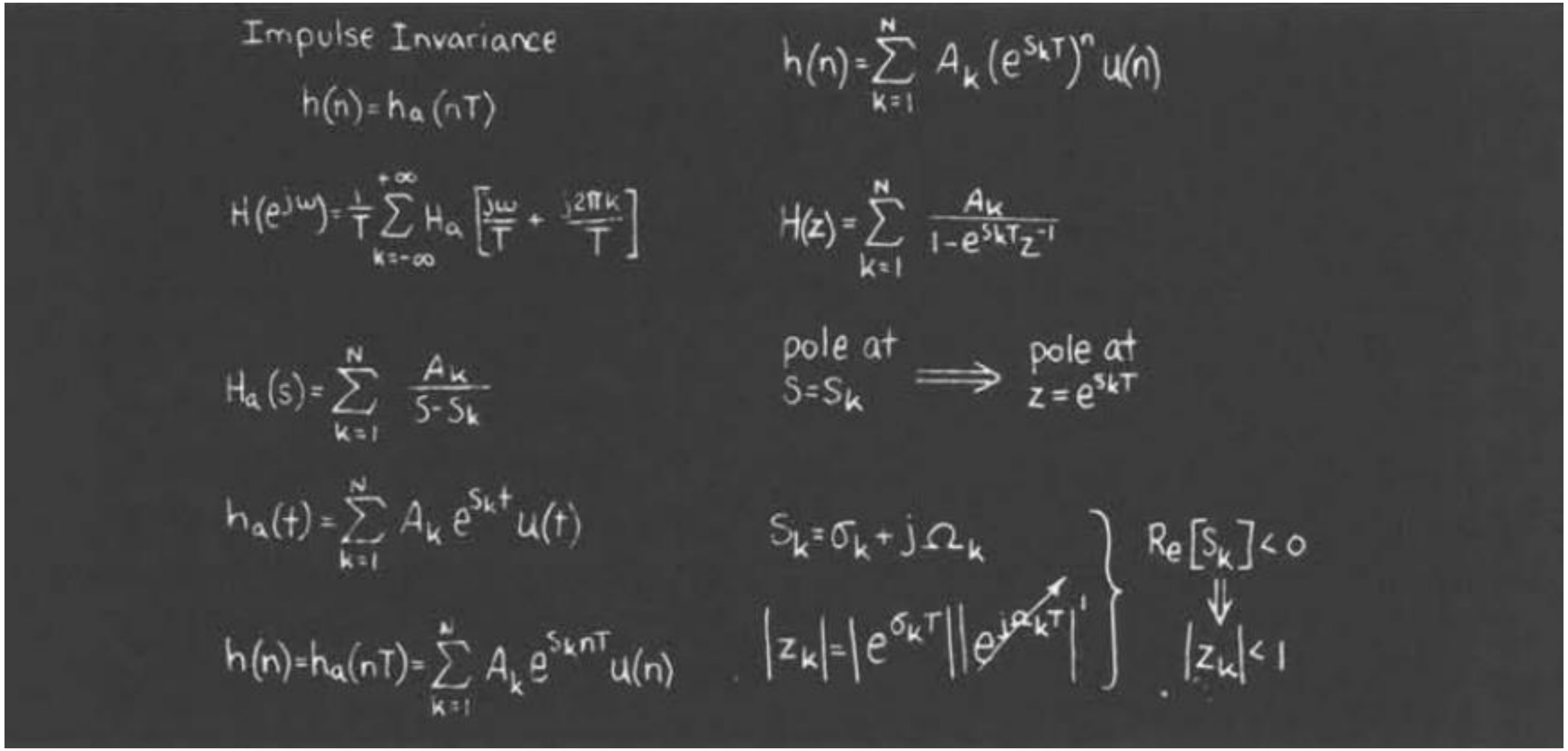
$\frac{dy_a(t)}{dt}\Big|_{t=nT} \longrightarrow \Delta^{(1)}[y(n)]$

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

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

$\mathcal{Z}\left[\frac{y(n+1)-y(n)}{T}\right] = \frac{z-1}{T} Y(z)$

Class Review



Impulse Invariance

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

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

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

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

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

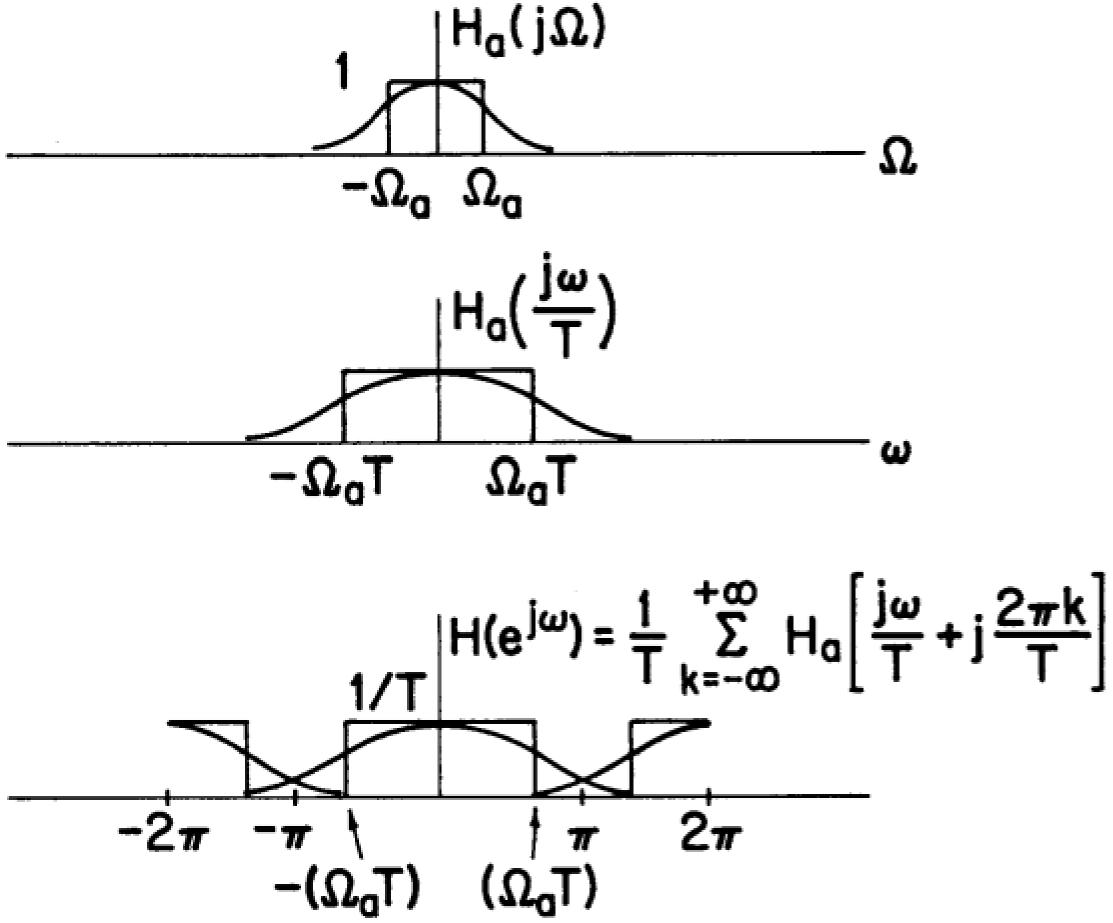
$$h(n) = \sum_{k=1}^{N} A_k (e^{s_k T})^n u(n)$$

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

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

$$\left.\begin{aligned} s_k &= \sigma_k + j\Omega_k \\ |z_k| &= |e^{\sigma_k T}|\,\cancelto{1}{|e^{j\Omega_k T}|} \end{aligned}\right\} \quad \begin{aligned} &\operatorname{Re}[s_k] < 0 \\ &\quad\Downarrow \\ &|z_k| < 1 \end{aligned}$$

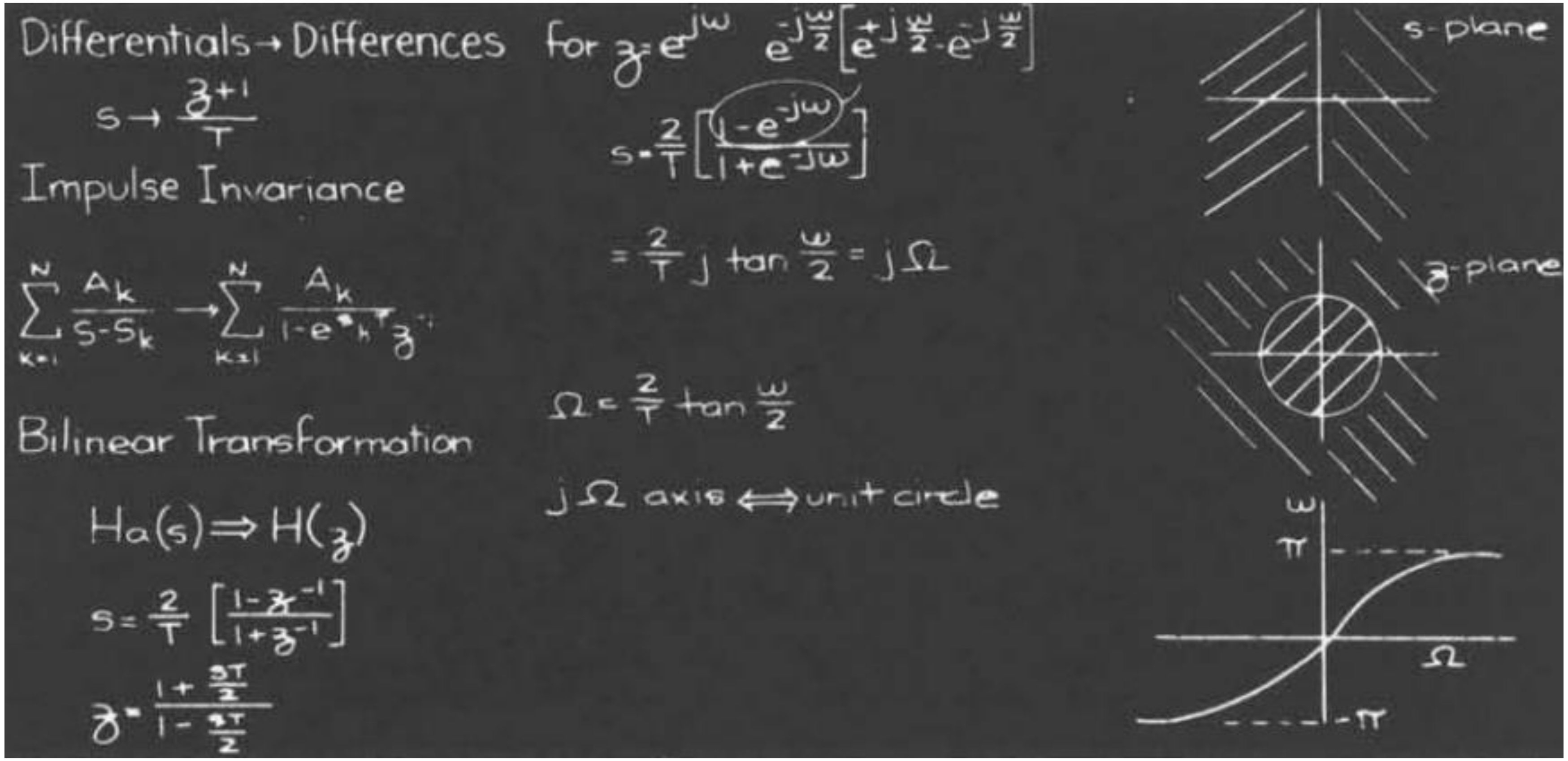
Class Review



$H_a(j\Omega)$
1
Ω
$-\Omega_a$
Ω_a
$H_a\left(\frac{j\omega}{T}\right)$
ω
$-\Omega_a T$
$\Omega_a T$
$H(e^{j\omega}) = \frac{1}{T}\sum_{k=-\infty}^{+\infty} H_a\left[\frac{j\omega}{T} + j\frac{2\pi k}{T}\right]$
1/T
-2π
$-\pi$
π
2π
$-(\Omega_a T)$
$(\Omega_a T)$

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

Class Review



Differentials → Differences

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

Impulse Invariance

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

Bilinear Transformation

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

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

$$z = \frac{1+\frac{sT}{2}}{1-\frac{sT}{2}}$$

for $z = e^{j\omega}$ $\quad e^{-j\frac{\omega}{2}}\left[e^{+j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}\right]$

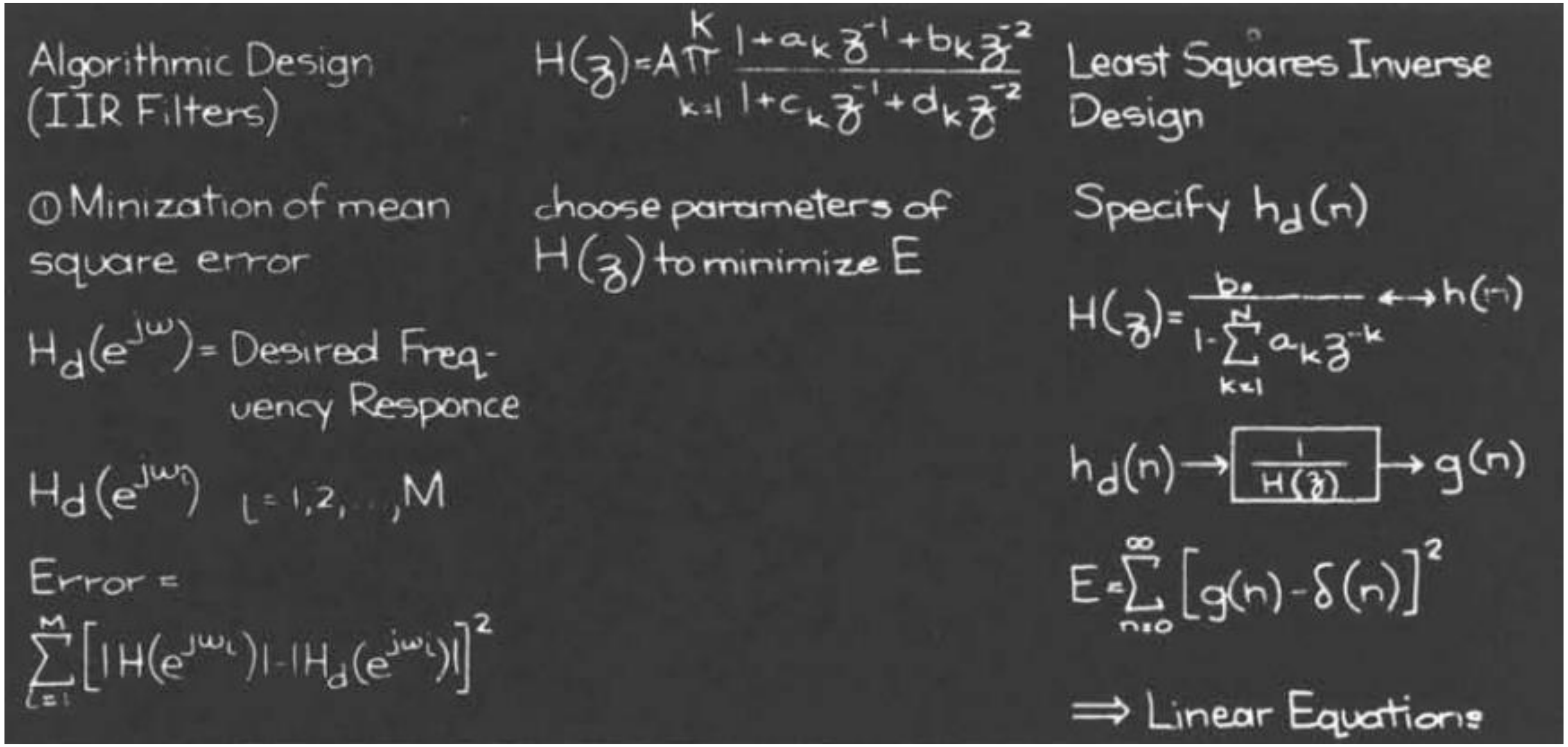
$$s = \frac{2}{T}\left[\frac{1-e^{-j\omega}}{1+e^{-j\omega}}\right]$$

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

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

$j\Omega$ axis $\Longleftrightarrow$ unit circle

Class Review



Algorithmic Design
(IIR Filters)

① Minization of mean square error

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

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

Error =

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

$$H(z)=A\prod_{k=1}^{K}\frac{1+a_kz^{-1}+b_kz^{-2}}{1+c_kz^{-1}+d_kz^{-2}}$$

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

Least Squares Inverse Design

Specify $h_d(n)$

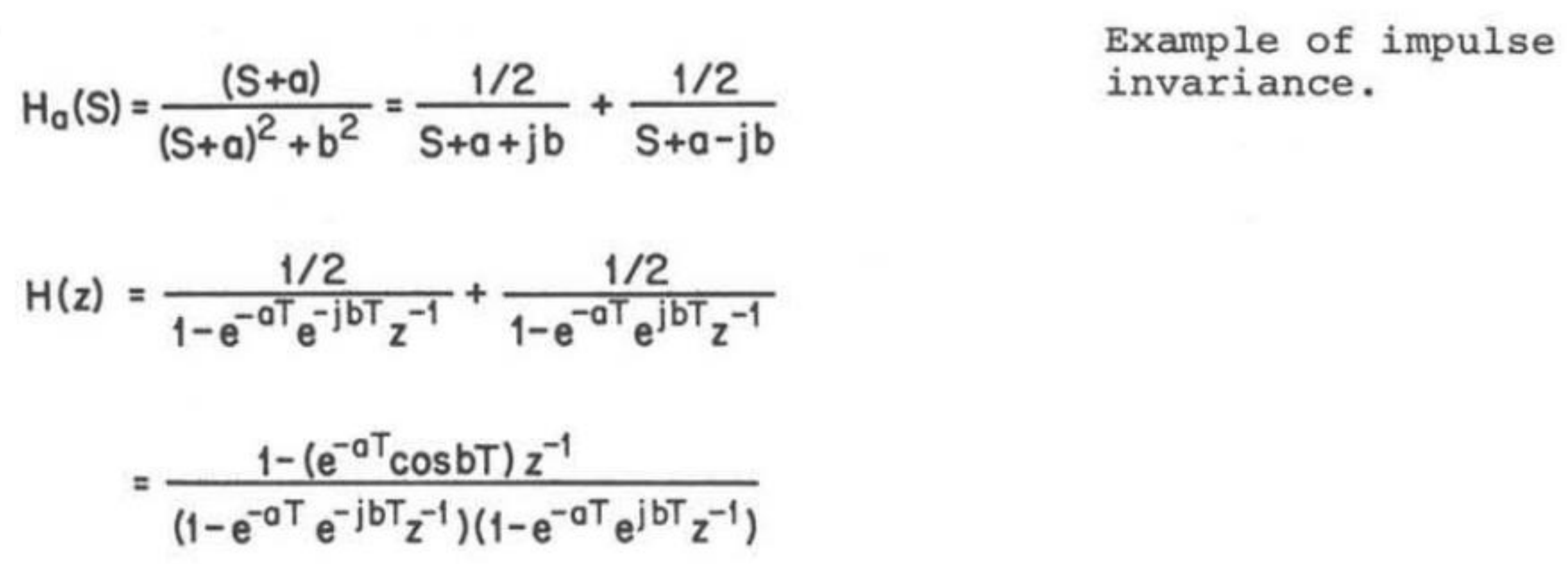
$$H(z)=\frac{b_0}{1-\sum_{k=1}^{N}a_kz^{-k}} \longleftrightarrow h(n)$$

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

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

$\Longrightarrow$ Linear Equations

Class Review



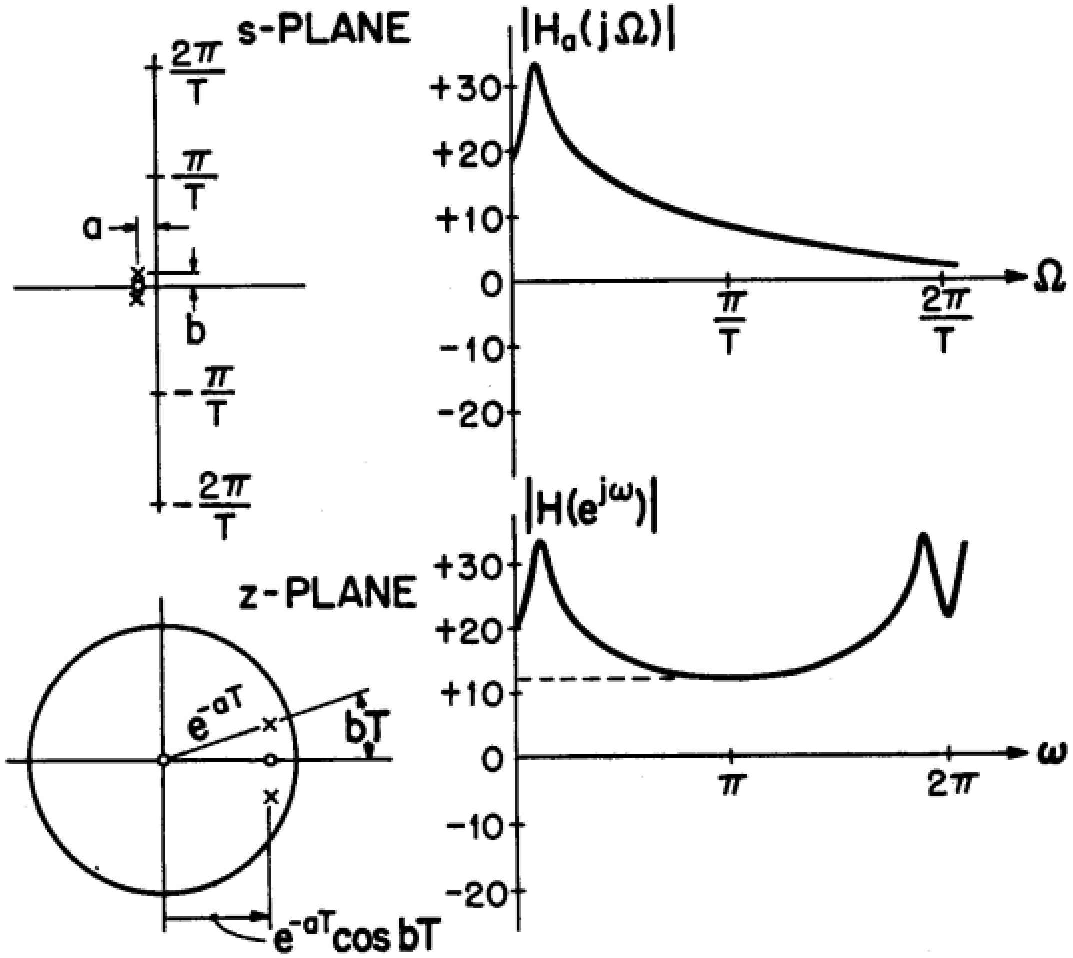
$$H_a(S) = \frac{(S+a)}{(S+a)^2+b^2} = \frac{1/2}{S+a+jb} + \frac{1/2}{S+a-jb}$$

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

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

Example of impulse invariance.

Class Review



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

Class Review

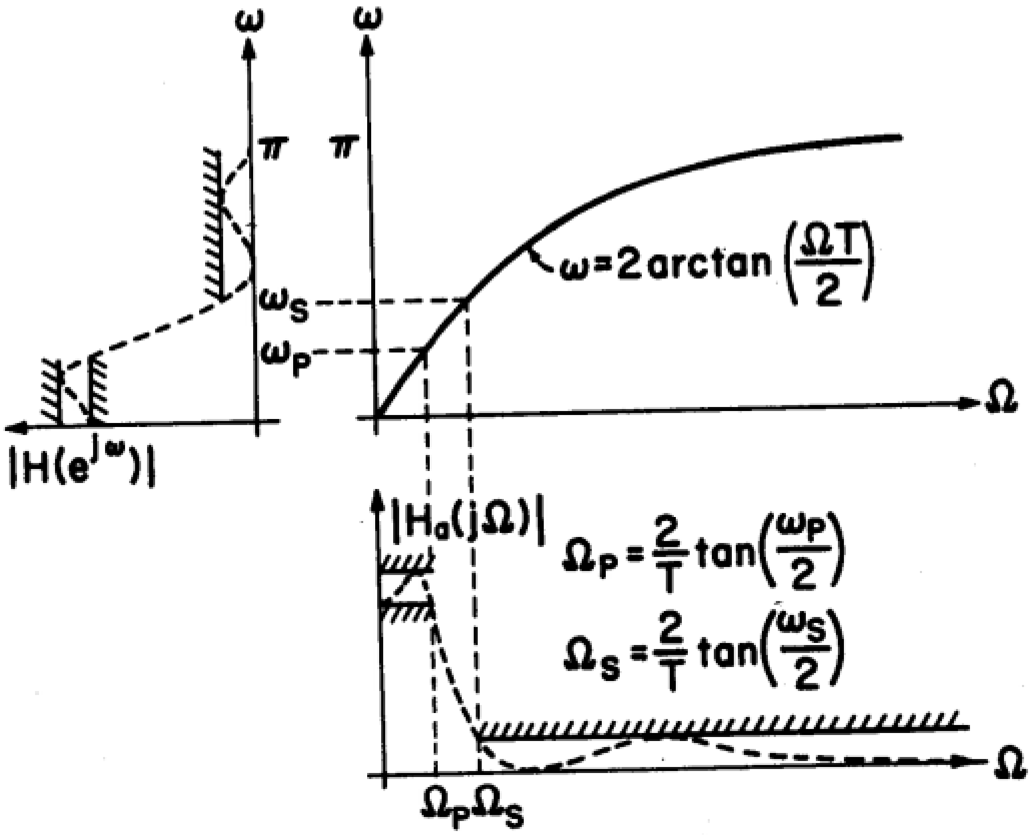


Illustration of effect of frequency warping inherent in the bilinear transformation.

Class Review

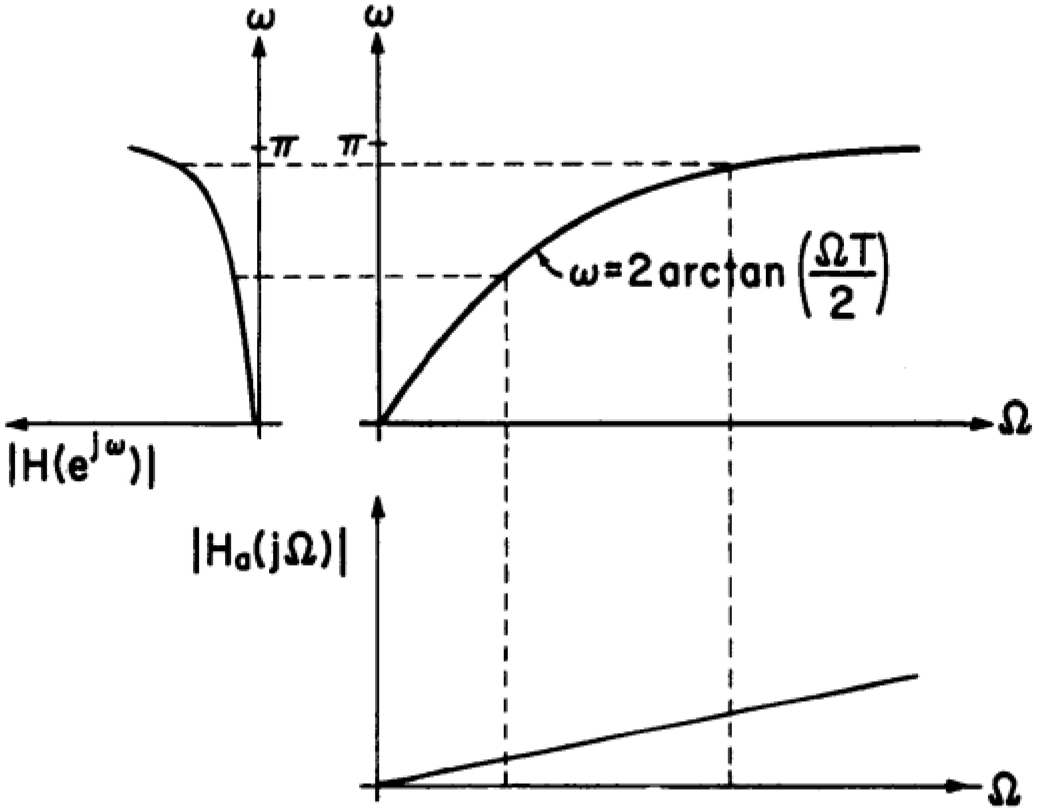
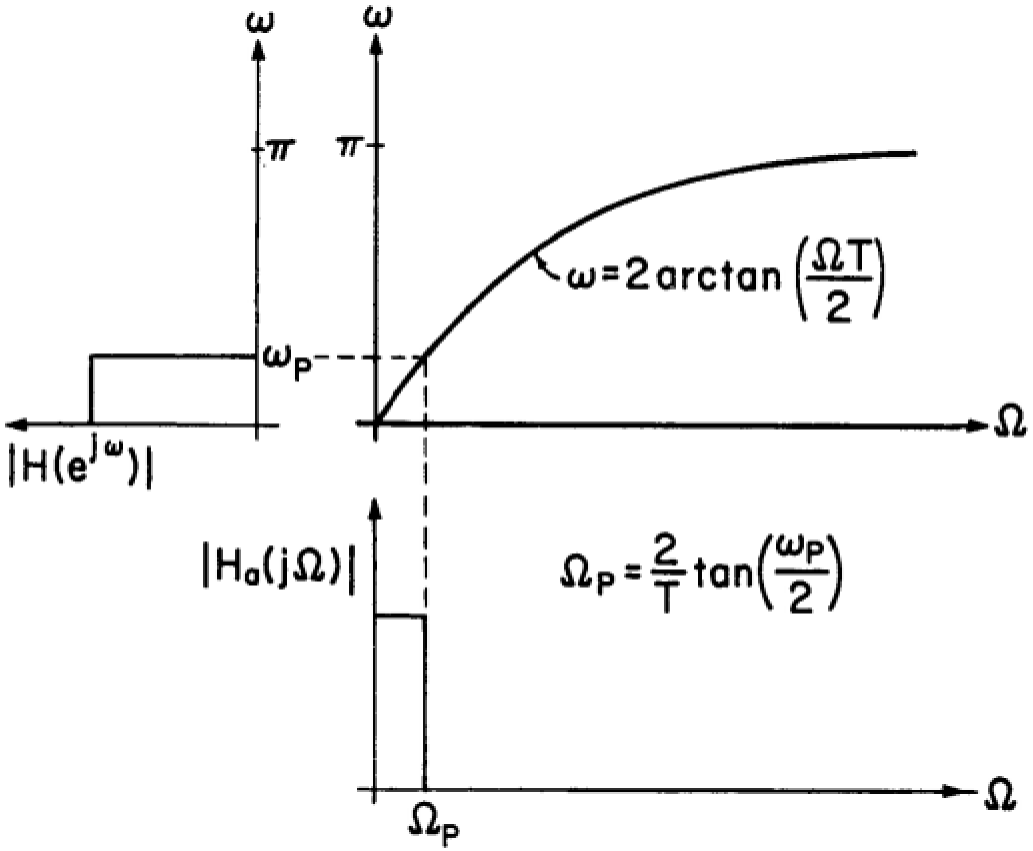


Illustration of effect of bilinear transformation on a piecewise constant frequency response characteristic.

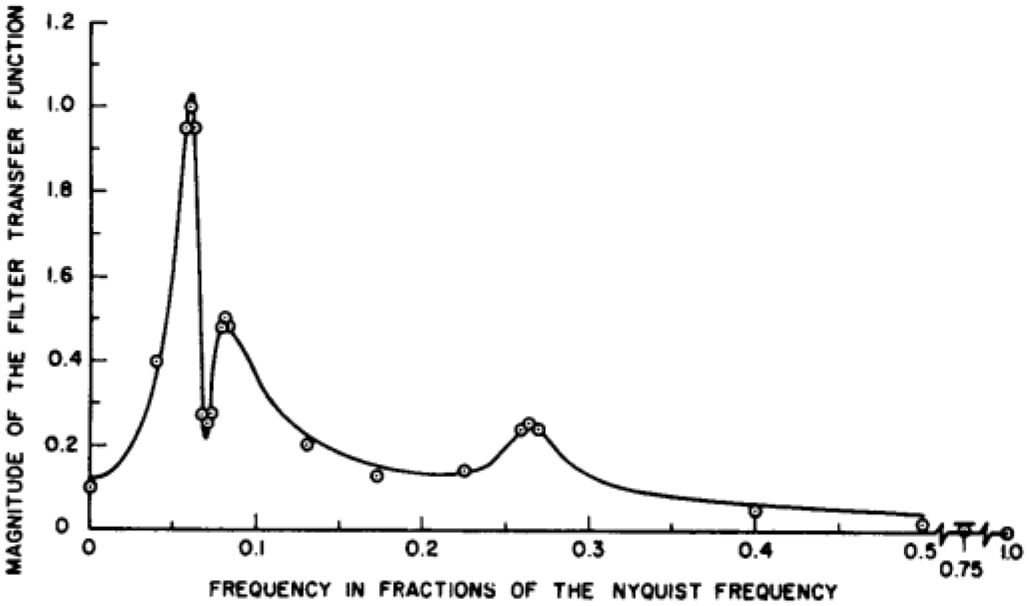
Class Review



ω
ω
π
π
$\omega = 2\arctan\left(\frac{\Omega T}{2}\right)$
ω_P
Ω
$|H(e^{j\omega})|$
$|H_a(j\Omega)|$
$\Omega_P = \frac{2}{T}\tan\left(\frac{\omega_P}{2}\right)$
Ω
Ω_P

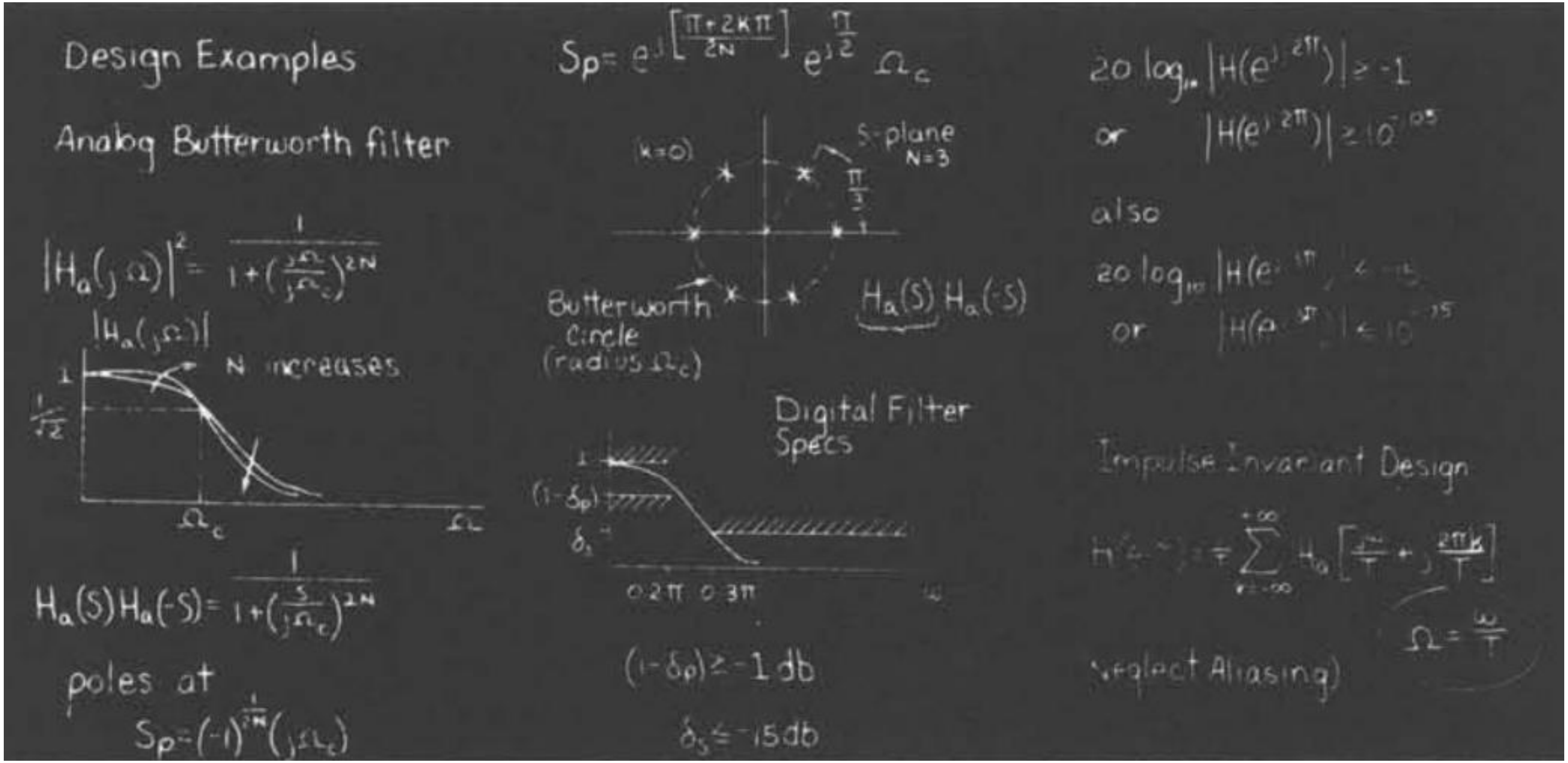
Illustration of effect of bilinear transformation on an equiripple frequency response characteristic.

Class Review



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

Class Review



Design Examples

Analog Butterworth filter

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

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

poles at

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

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

$$\underbrace{H_a(S)}\,H_a(-S)$$

Digital Filter Specs

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

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

$$20\log_{10}|H(e^{j.2\pi})| \geq -1$$

or $|H(e^{j.2\pi})| \geq 10^{-.05}$

also

$$20\log_{10}|H(e^{j.3\pi})| \leq -15$$

or $|H(e^{j.3\pi})| \leq 10^{-.75}$

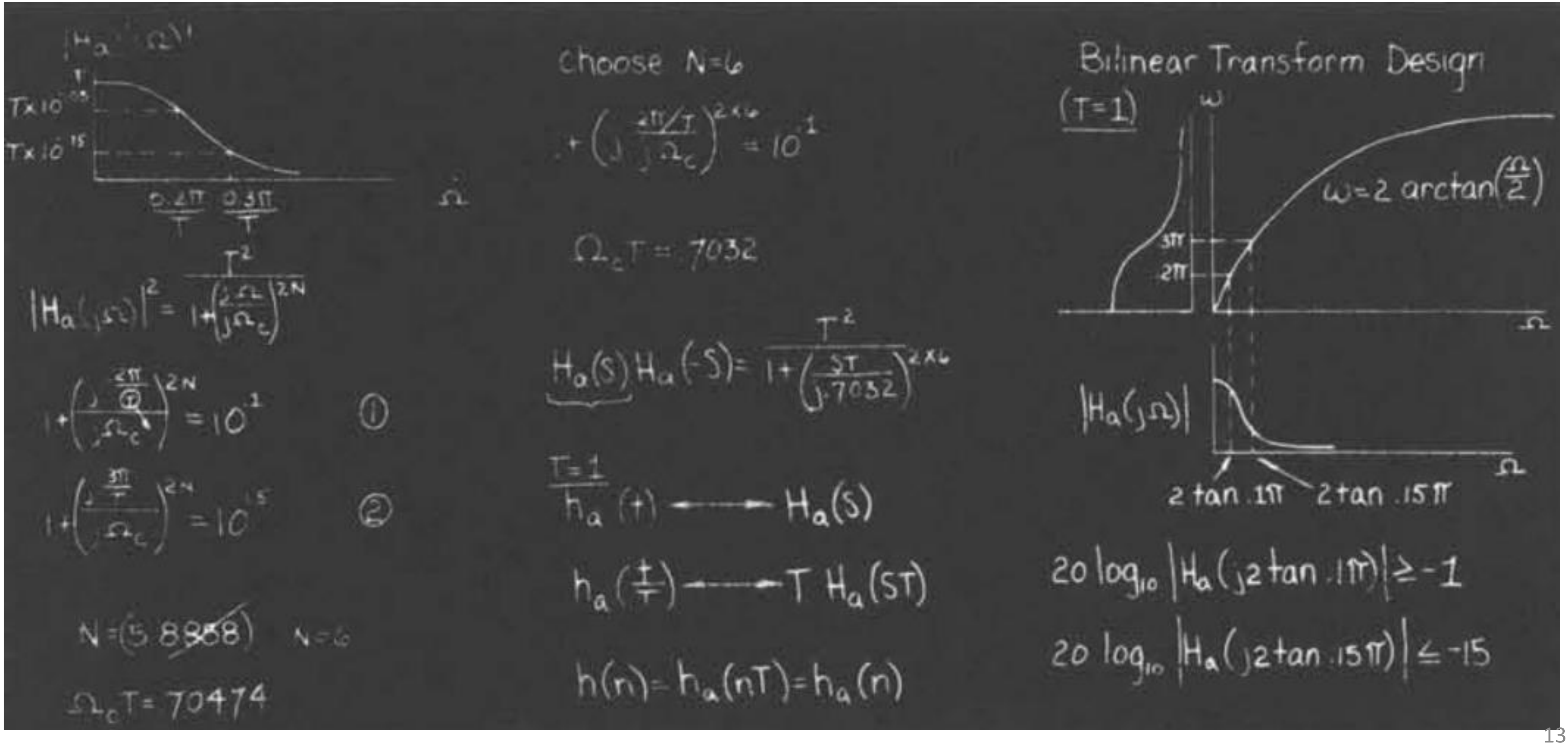
Impulse Invariant Design

$$H(e^{j\omega}) = \frac{1}{T}\sum_{k=-\infty}^{+\infty} H_a\left[\frac{j\omega}{T} + j\frac{2\pi k}{T}\right]$$

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

(neglect Aliasing)

Class Review



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

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

$$1+\left(\frac{j\frac{.3\pi}{T}}{j\Omega_c}\right)^{2N} = 10^{1.5} \quad ②$$

$$N = (5.8\cancel{858}) \quad N=6$$

$$\Omega_c T = .70474$$

Choose N=6

$$1+\left(\frac{j\,.2\pi/T}{j\,\Omega_c}\right)^{2\times6} = 10^{.1}$$

$$\Omega_c T = .7032$$

$$\underbrace{H_a(S)}\,H_a(-S) = \frac{T^2}{1+\left(\frac{ST}{j\,.7032}\right)^{2\times6}}$$

$\underline{T=1}$

$$h_a(t) \longleftrightarrow H_a(S)$$

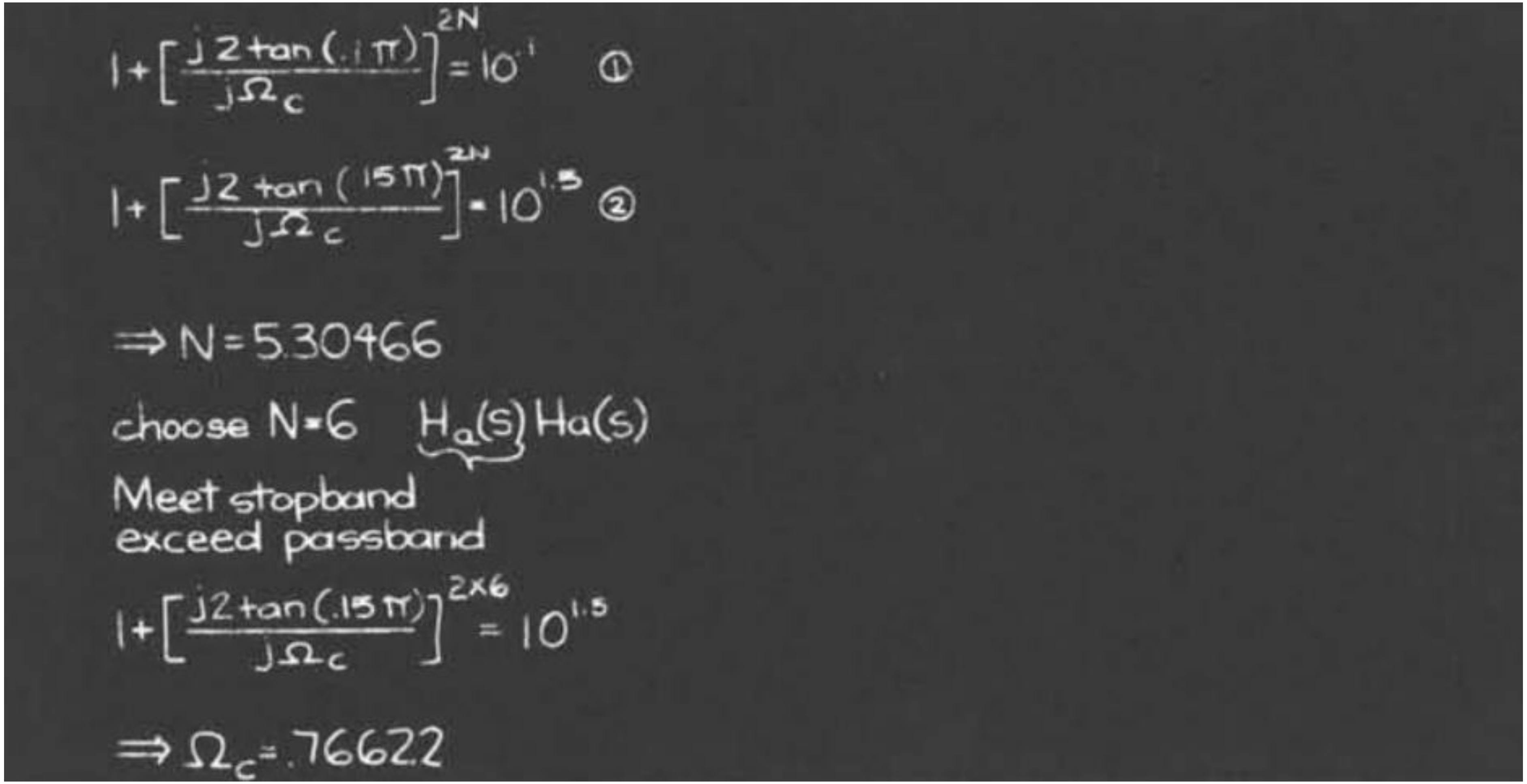
$$h_a\left(\frac{t}{T}\right) \longleftrightarrow T\,H_a(ST)$$

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

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

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

Class Review



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

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

$\Rightarrow N=5.30466$

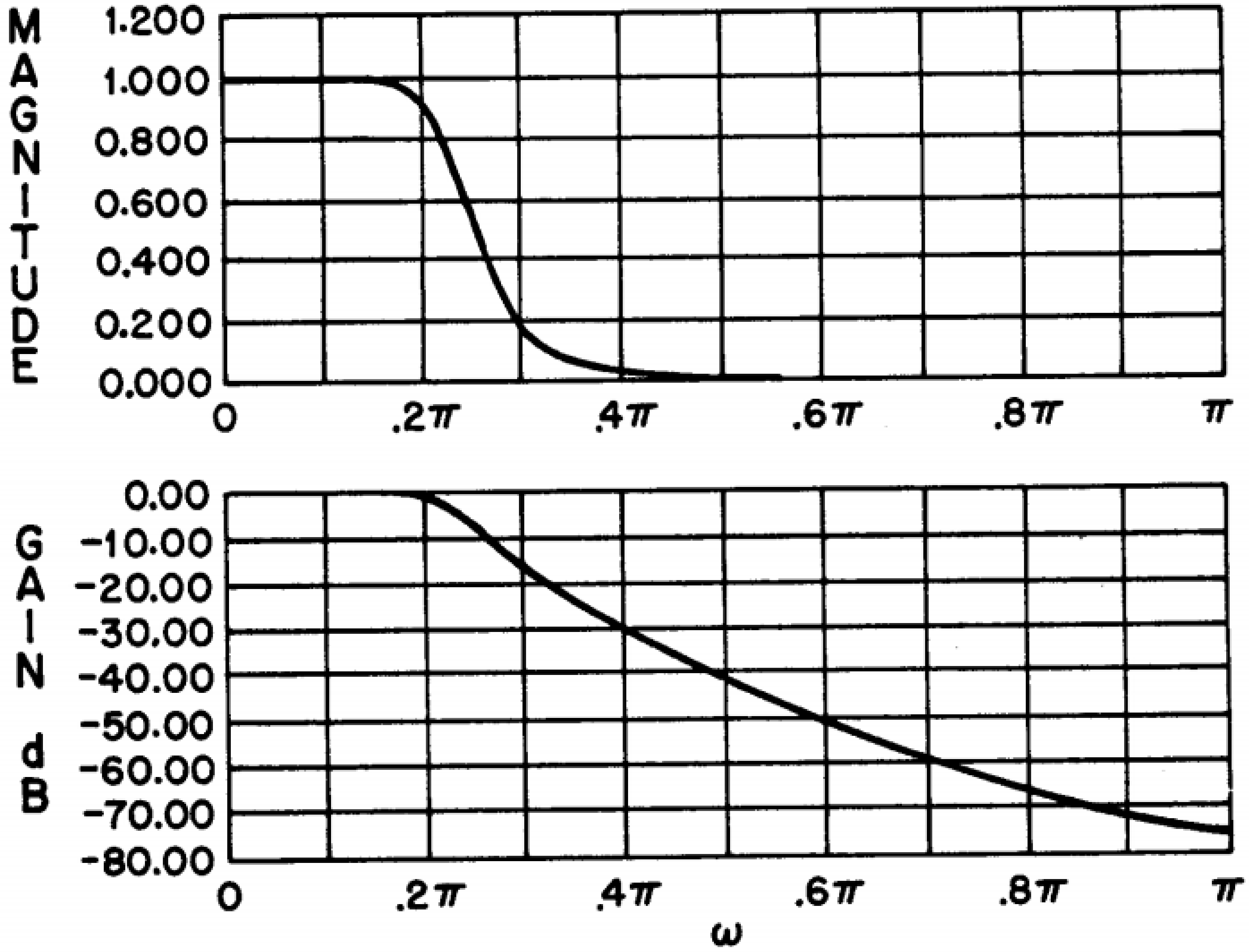
choose $N=6$ $\quad \underbrace{H_a(s)}\,Ha(s)$

Meet stopband
exceed passband

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

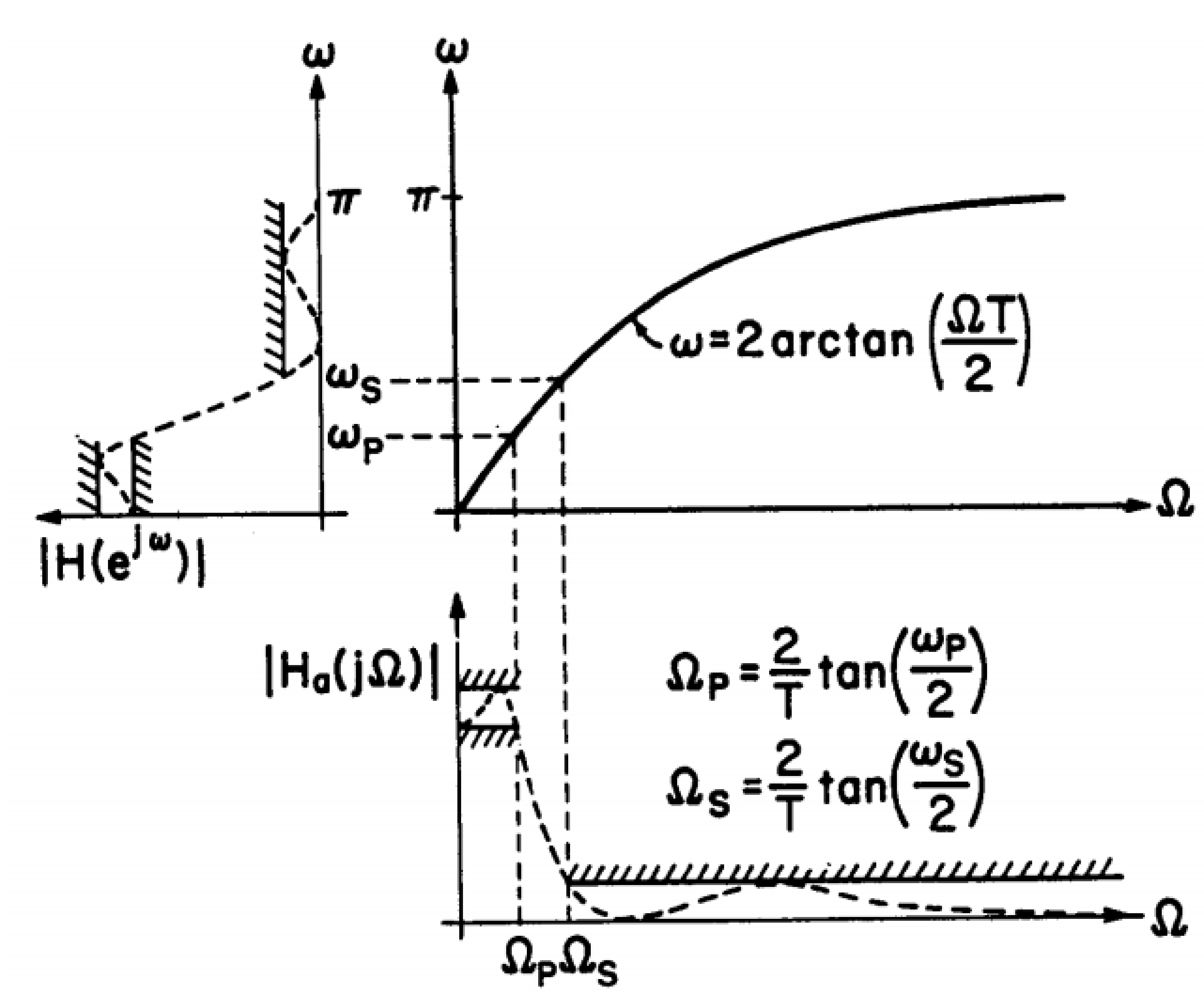
$\Rightarrow \Omega_c=.76622$

Class Review



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

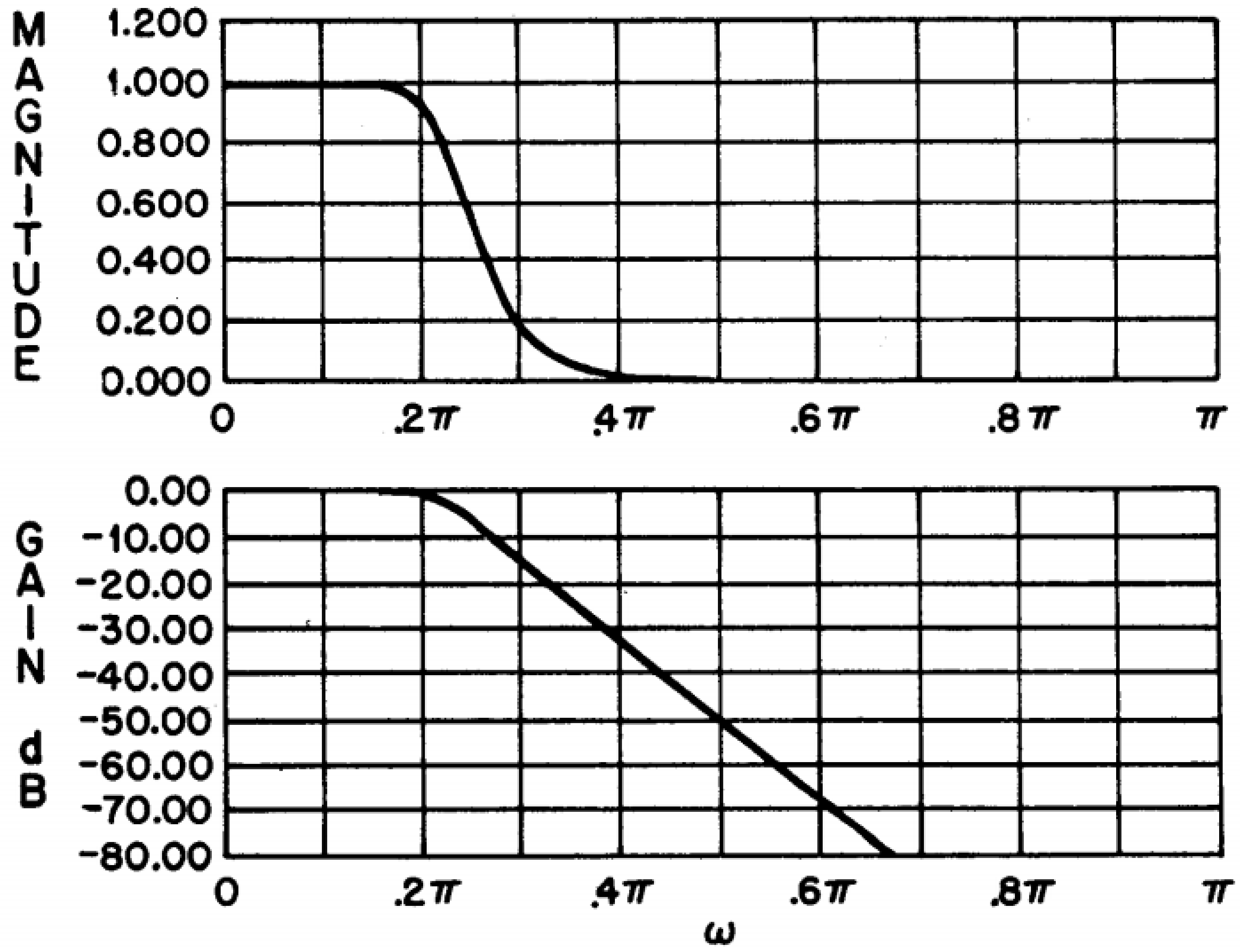
Class Review



ω
ω
π
π
ω_S
ω_P
Ω
$|H(e^{j\omega})|$
$\omega = 2\arctan\left(\frac{\Omega T}{2}\right)$
$|H_a(j\Omega)|$
$\Omega_P = \frac{2}{T}\tan\left(\frac{\omega_P}{2}\right)$
$\Omega_S = \frac{2}{T}\tan\left(\frac{\omega_S}{2}\right)$
Ω
$\Omega_P \Omega_S$

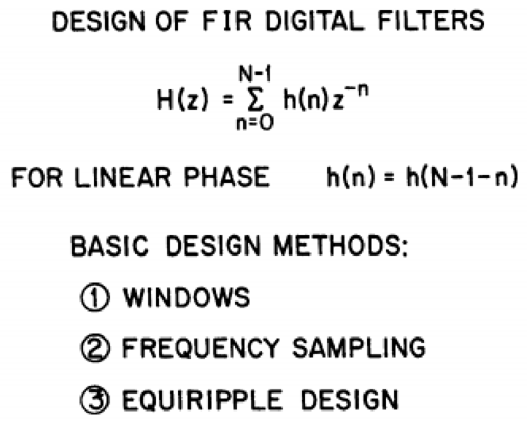
The bilinear transformation.

Class Review



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

Class Review



DESIGN OF FIR DIGITAL FILTERS

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

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

BASIC DESIGN METHODS:

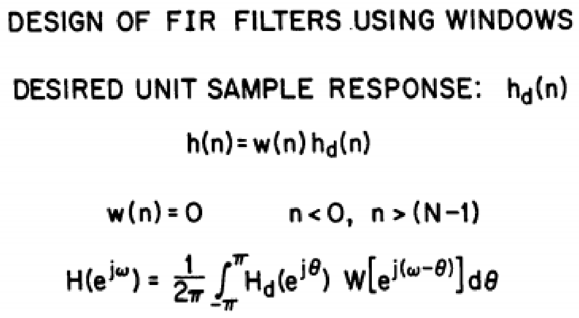
① WINDOWS

② FREQUENCY SAMPLING

③ EQUIRIPPLE DESIGN

Basic design methods for FIR digital filters

a.



DESIGN OF FIR FILTERS USING WINDOWS

DESIRED UNIT SAMPLE RESPONSE: $h_d(n)$

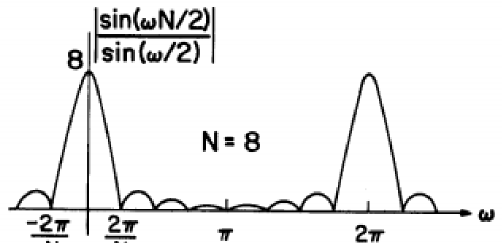
$$h(n) = w(n) h_d(n)$$

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

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

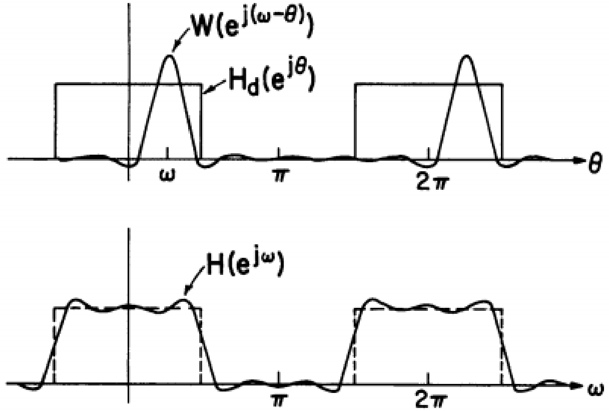
Design of FIR filters using the window method.

b.

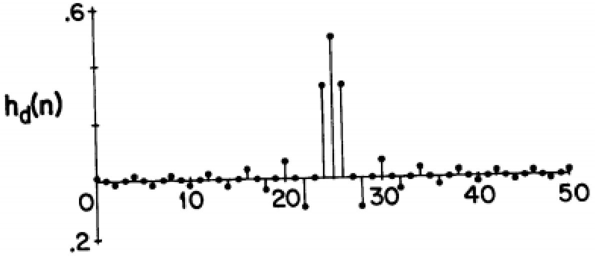


Magnitude of the Fourier transform for an eight point rectangular window.

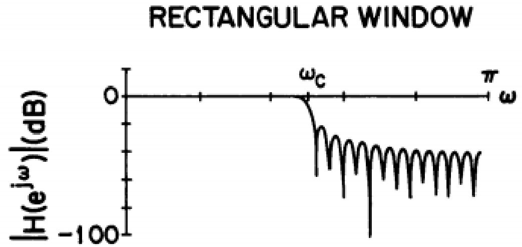
Class Review



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

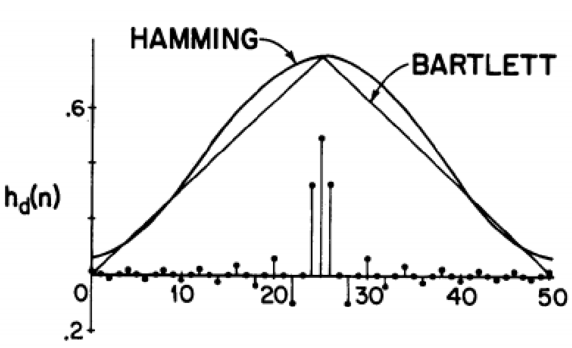


Unit-sample response of an ideal low-pass filter truncated by a 51-point rectangular window.

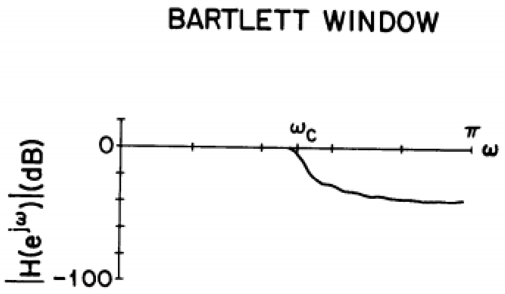


Frequency response corresponding to the unit-sample response in viewgraph e.

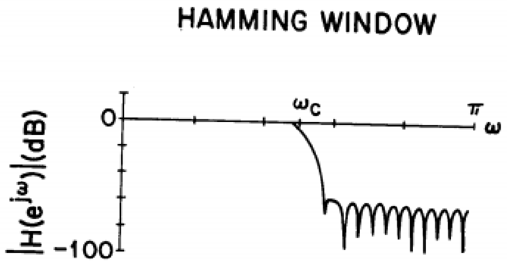
Class Review



The Hamming and Bartlett windows.

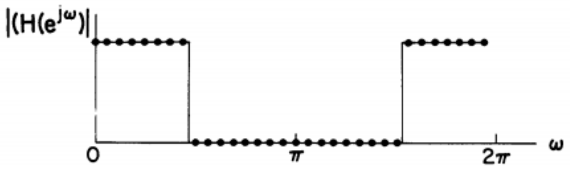


Frequency response of an FIR lowpass filter obtained by multiplying the unit-sample response of an ideal low pass filter by a Bartlett window.



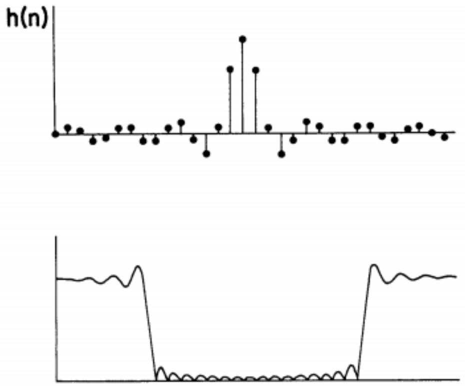
Frequency response of an FIR lowpass filter filter obtained by multiplying the unit sample response of an ideal lowpass filter by a Hamming window. (Note that the stopband attenuation is approximately 65 db <u>not</u> 30 db as stated in the lecture.)

Class Review

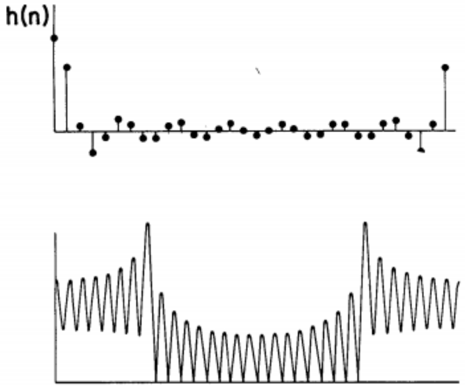


$$H(e^{j\omega_k}) = H_D(e^{j\omega_k}) \quad \omega_k = \frac{2\pi}{N}k \quad k = 0,1,\ldots,(N-1)$$

Equally spaced frequency samples of an ideal low pass filter.

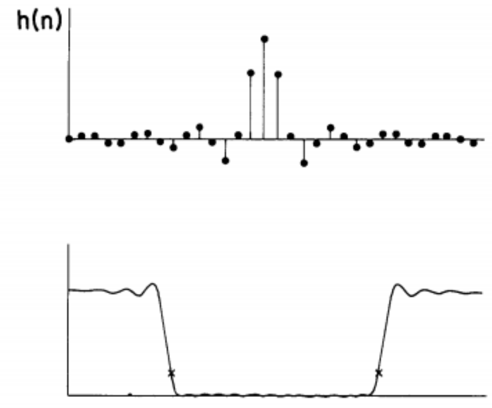


A unit-sample response the magnitude of whose DFT is equal to the frequency samples in viewgraph j.. The bottom trace is the magnitude of the Fourier transform of this unit-sample response.

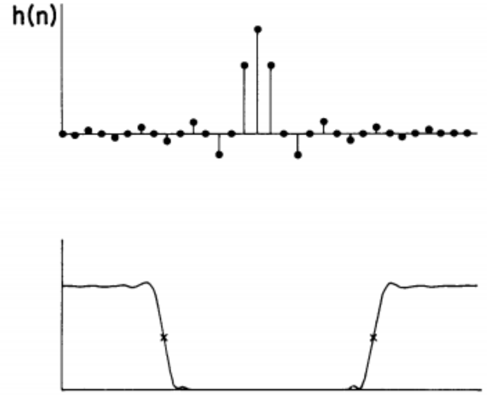


Another unit-sample response the magnitude of whose DFT is equal to the frequency samples in viewgraph j.. The bottom trace is the magnitude of the Fourier transform of this unit-sample response.

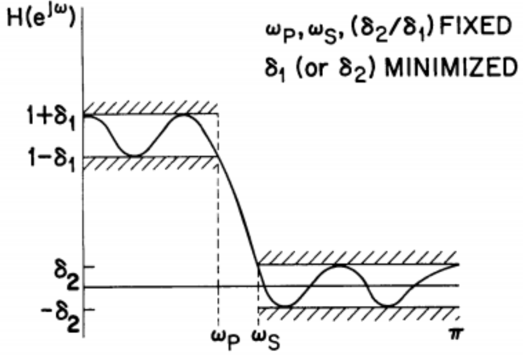
Class Review



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

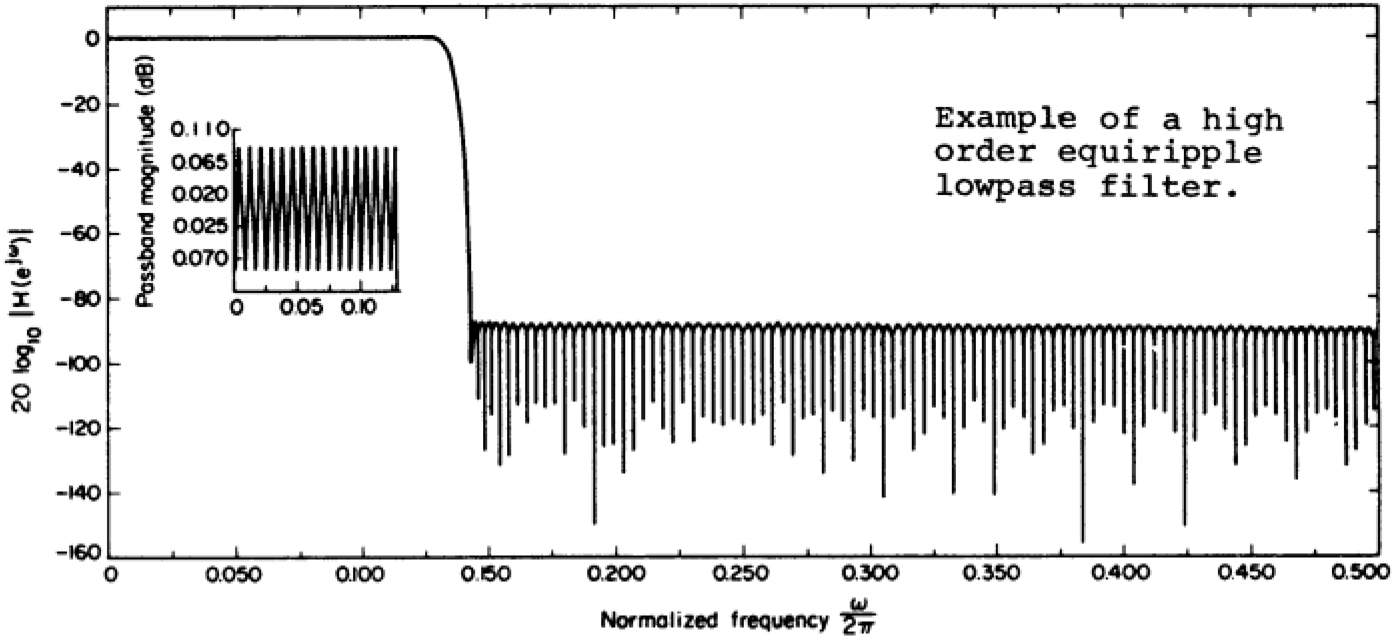


Similar to viewgraph
m. with a different
value of the frequenc
sample in the
transition band.



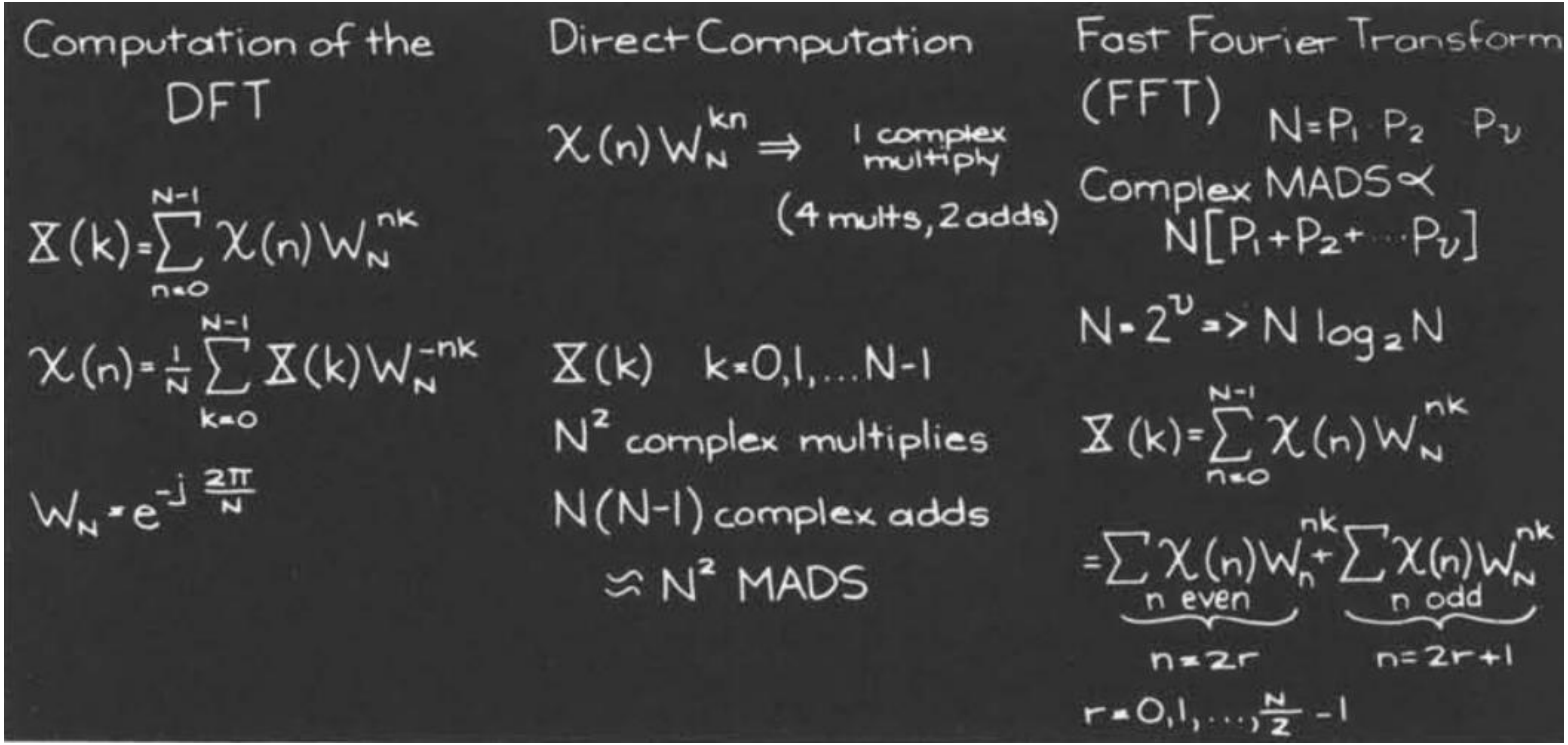
Equiripple approxi-
mation of a lowpass
filter.

Class Review



Example of a high order equiripple lowpass filter.

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

$x(n)\,W_N^{kn} \Rightarrow$ 1 complex multiply

(4 mults, 2 adds)

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

N^2 complex multiplies

$N(N-1)$ complex adds

$\approx N^2$ MADS

Fast Fourier Transform (FFT) $\quad N=P_1\cdot P_2 \quad P_\nu$

Complex MADS $\propto$

$N[P_1+P_2+\cdots P_\nu]$

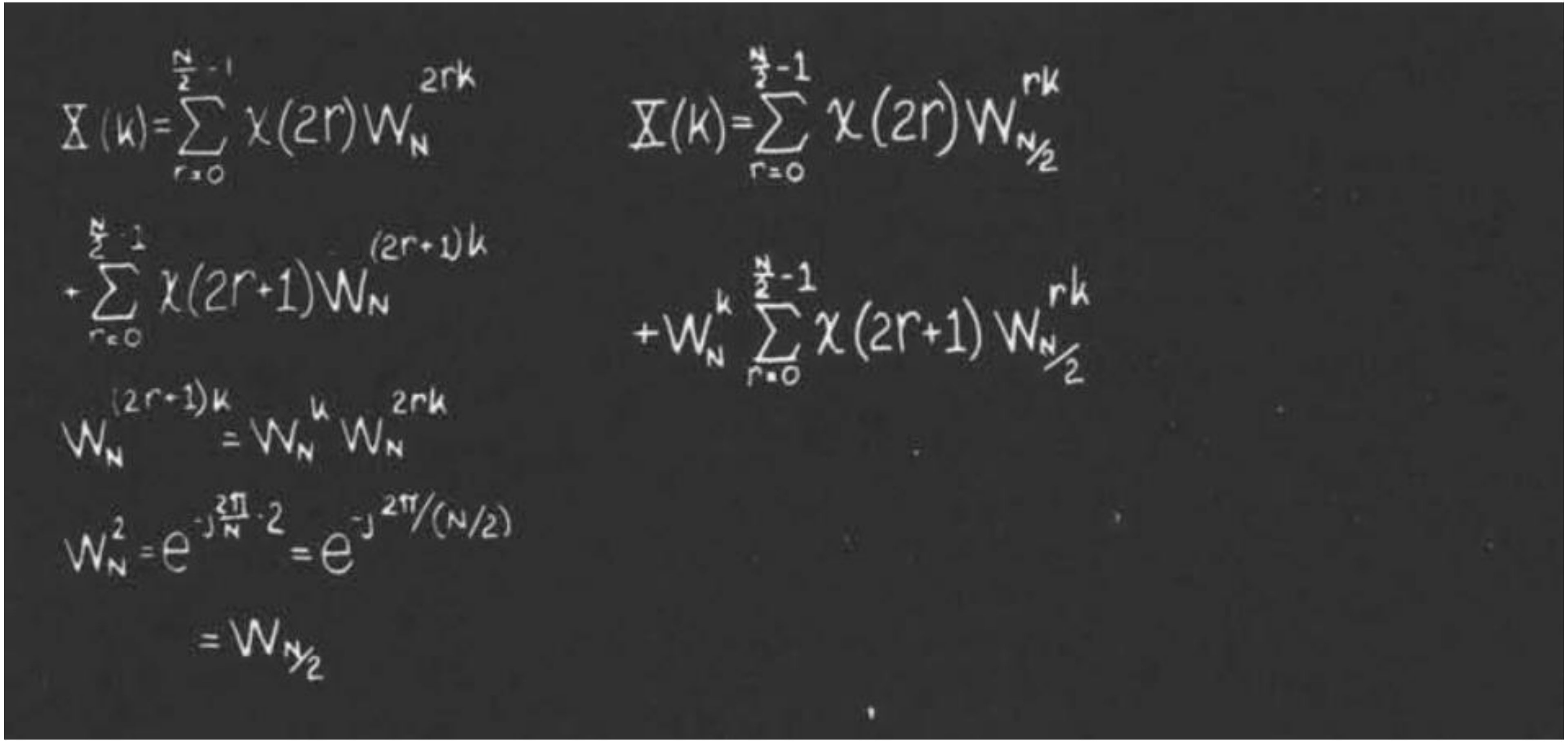
$N=2^\nu \Rightarrow N\log_2 N$

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

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

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

Class Review



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

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

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

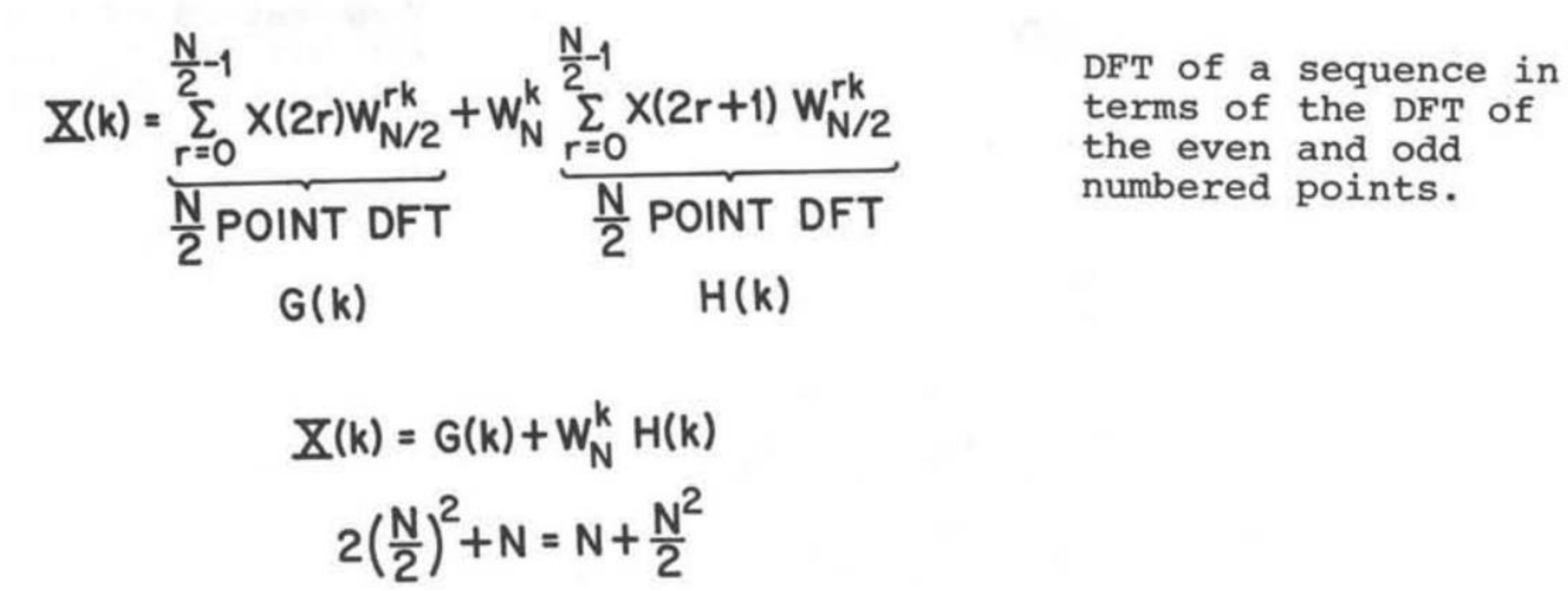
$$W_N^2=e^{-j\frac{2\pi}{N}\cdot 2}=e^{-j\,2\pi/(N/2)}$$

$$=W_{N/2}$$

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

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

Class Review



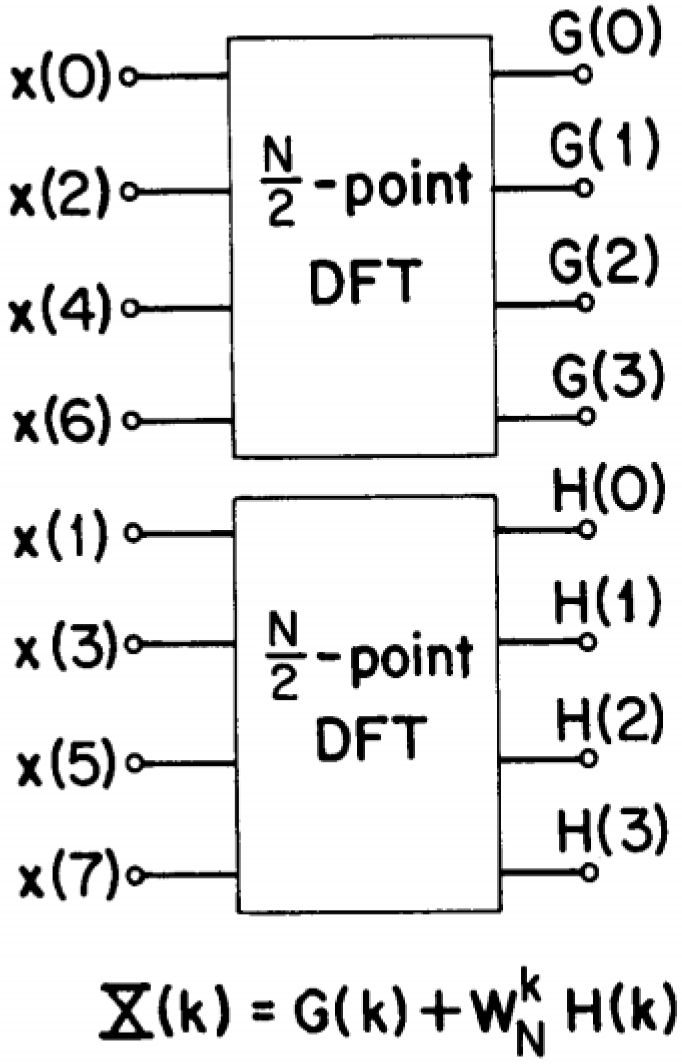
$$X(k) = \underbrace{\sum_{r=0}^{\frac{N}{2}-1} X(2r)W_{N/2}^{rk}}_{\substack{\frac{N}{2}\text{ POINT DFT} \\ G(k)}} + W_N^k \underbrace{\sum_{r=0}^{\frac{N}{2}-1} X(2r+1)\, W_{N/2}^{rk}}_{\substack{\frac{N}{2}\text{ POINT DFT} \\ H(k)}}$$

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

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

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

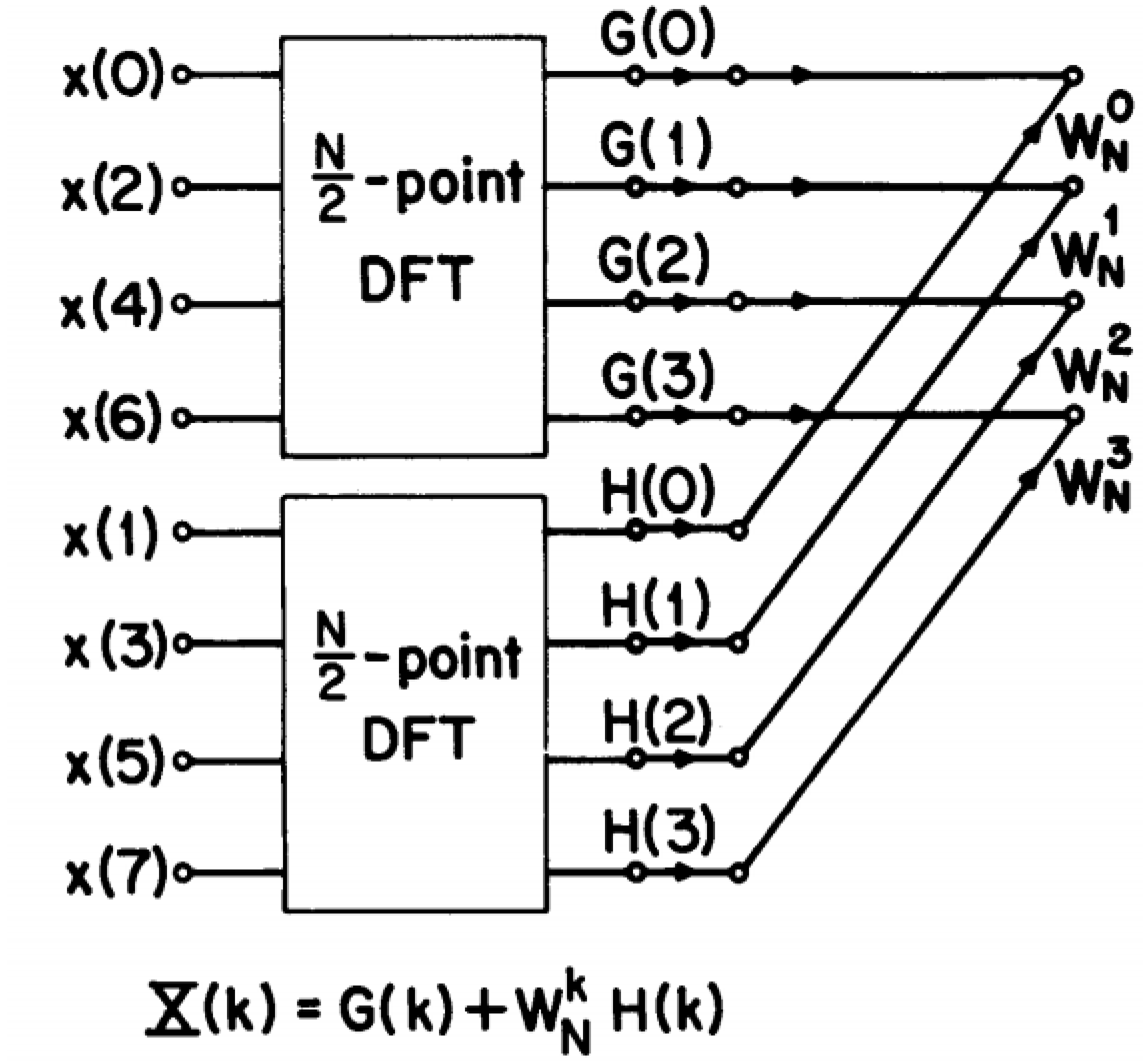
Class Review



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

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

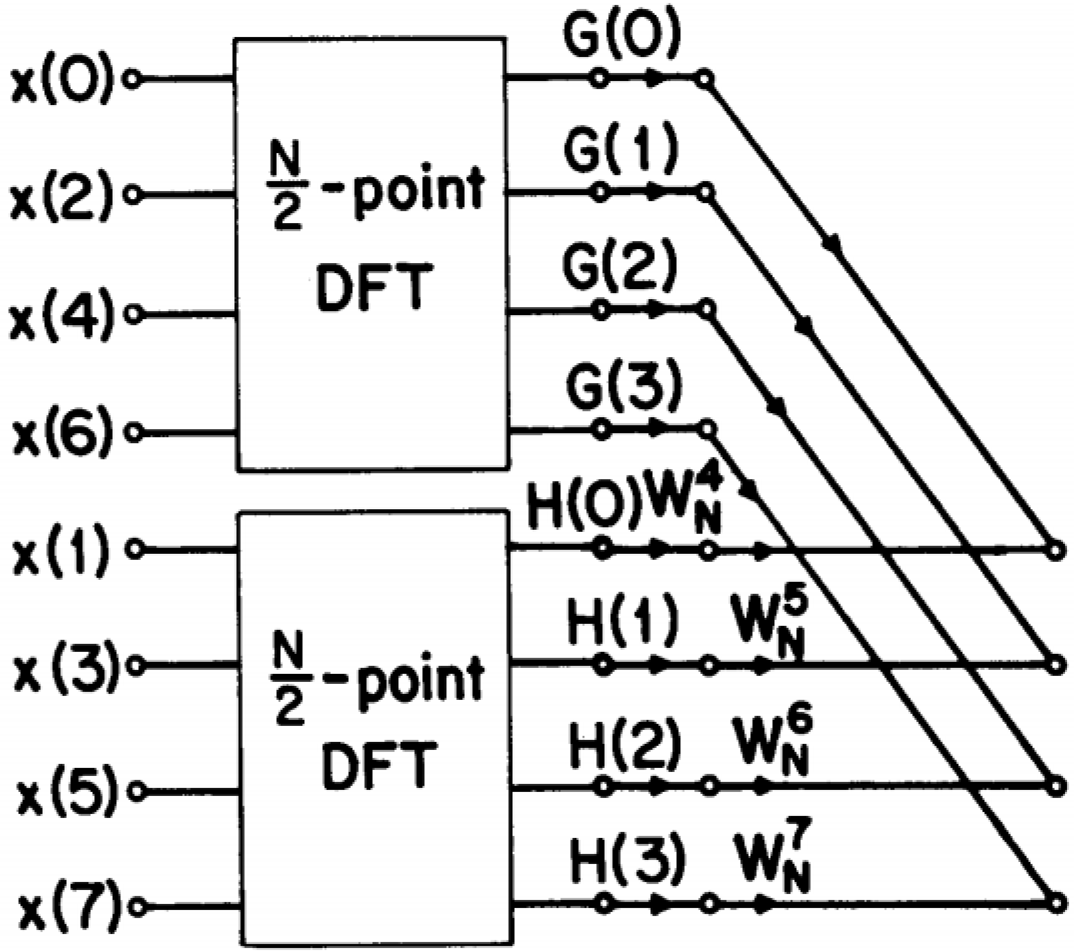
Class Review



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

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

Class Review

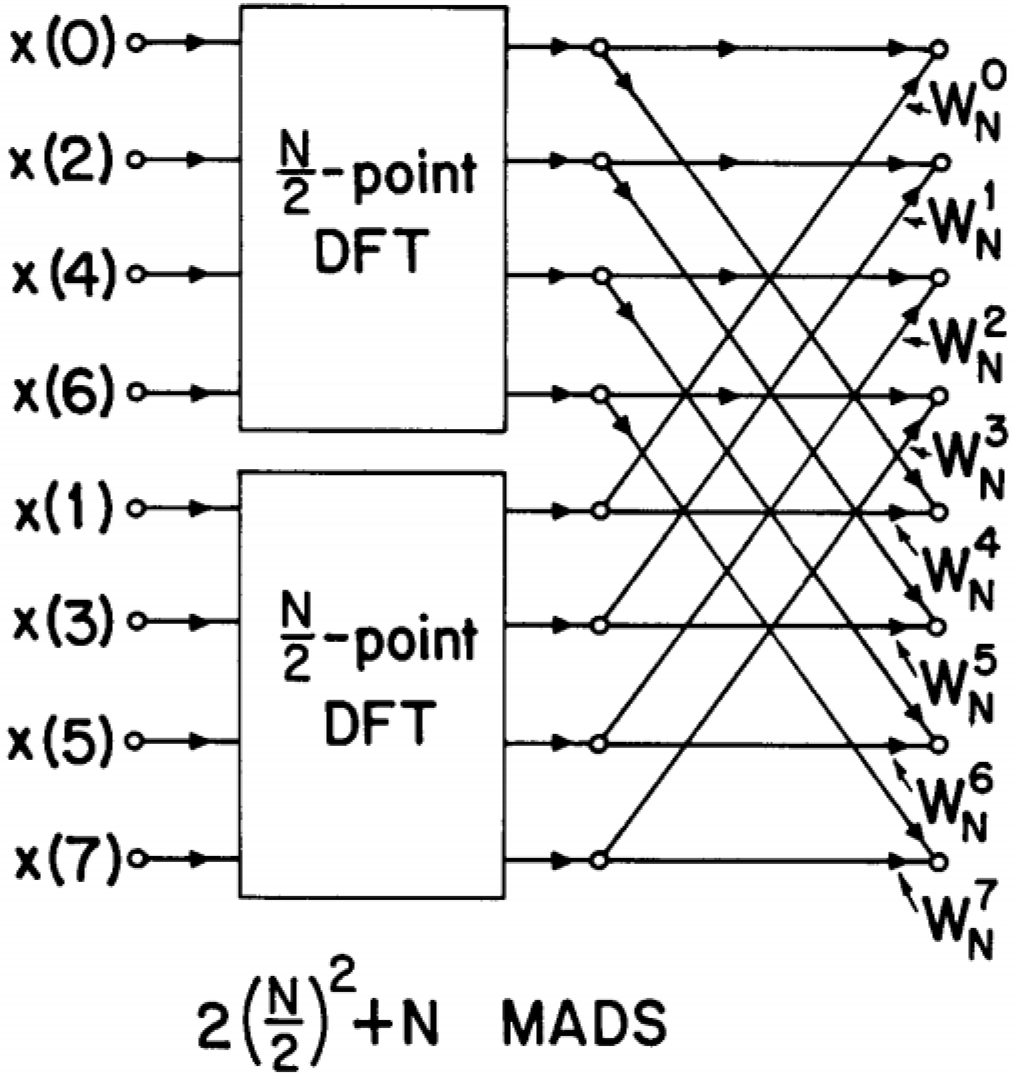


$$X(k) = G(k) + W_N^k H(k)$$

$$G(k+4) = G(k)$$
$$H(k+4) = H(k)$$

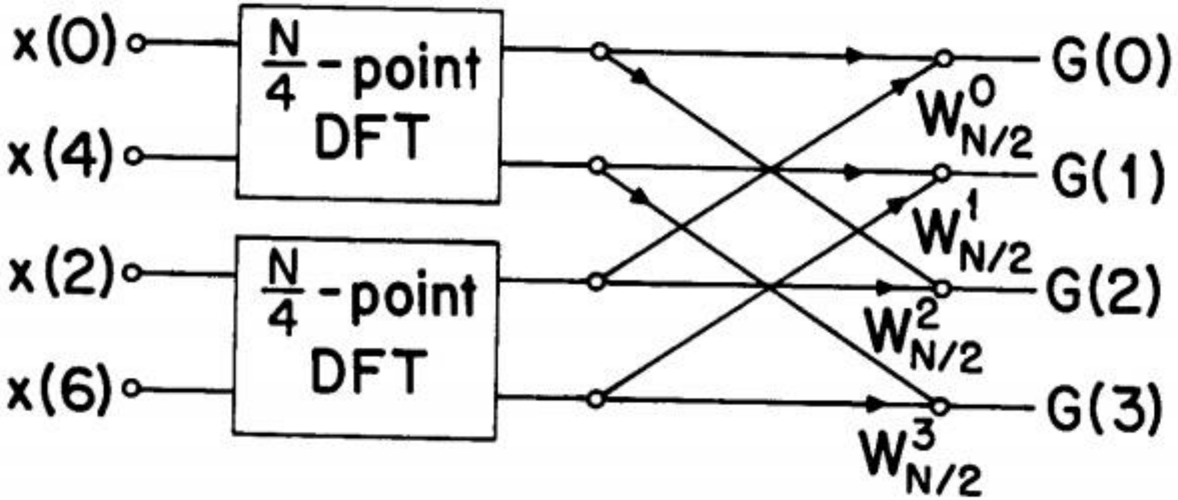
Combination of G(k) and H(k) to obtain second half of X(k)

Class Review



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

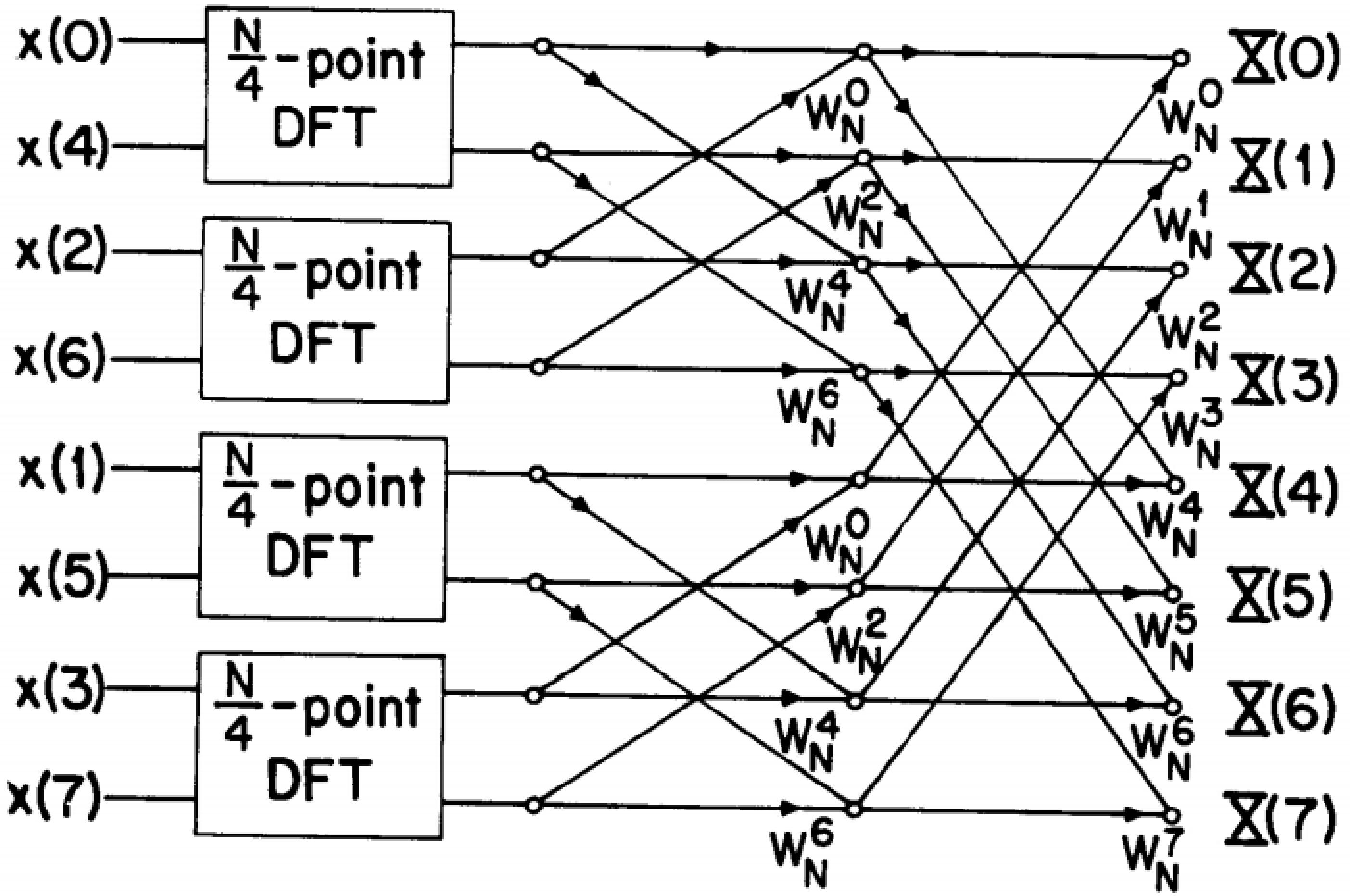
Class Review



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

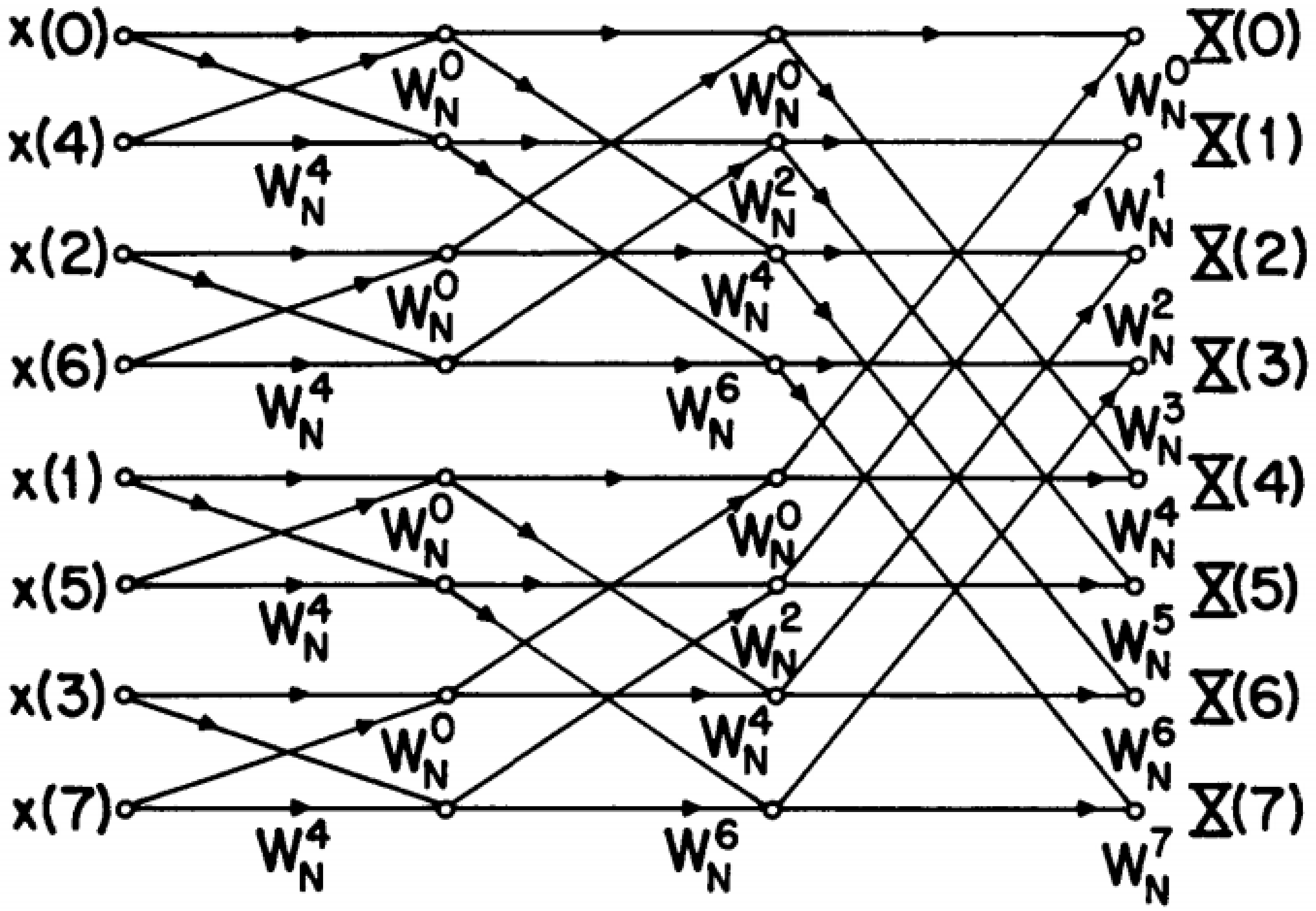
$$W_{\frac{N}{2}} = W_N^2$$

Class Review



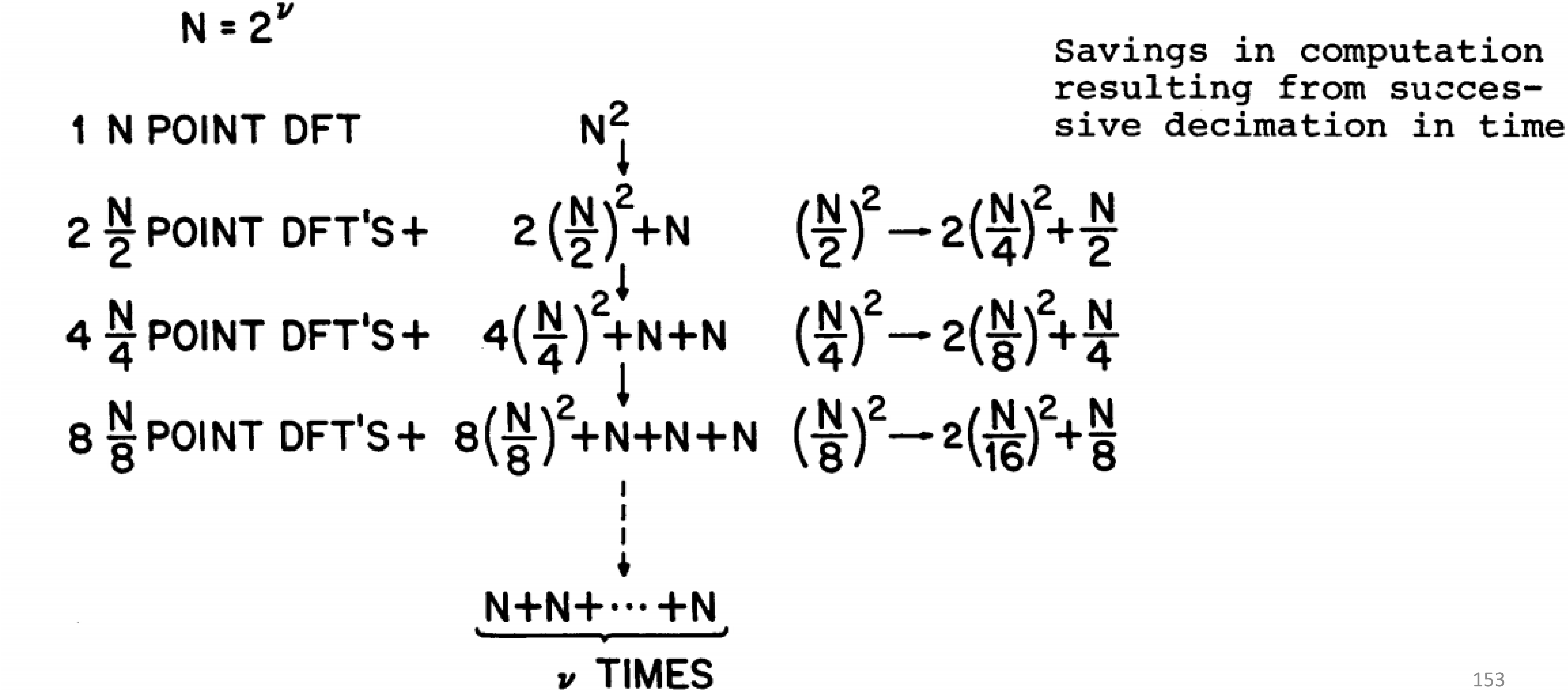
Computation of X (k) by combining flow-graph g and h.

Class Review



Flow-graph for computation of X(k)

Class Review

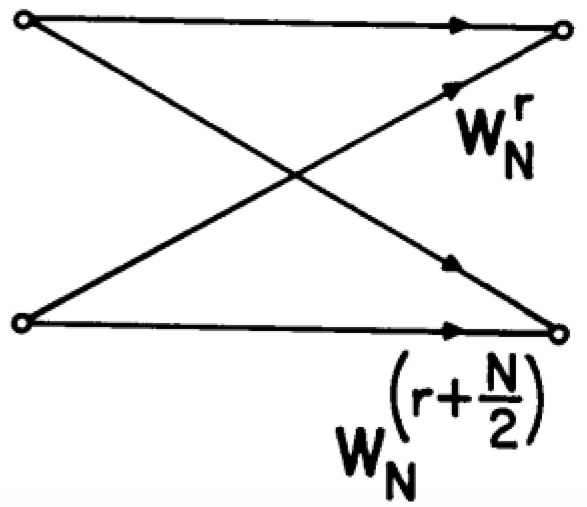


$N = 2^{\nu}$

Savings in computation resulting from successive decimation in time

$$
\begin{array}{lll}
1\ N \text{ POINT DFT} & N^2 & \\
 & \downarrow & \\
2\ \frac{N}{2} \text{ POINT DFT'S} + & 2\left(\frac{N}{2}\right)^2 + N & \left(\frac{N}{2}\right)^2 \longrightarrow 2\left(\frac{N}{4}\right)^2 + \frac{N}{2} \\
 & \downarrow & \\
4\ \frac{N}{4} \text{ POINT DFT'S} + & 4\left(\frac{N}{4}\right)^2 + N + N & \left(\frac{N}{4}\right)^2 \longrightarrow 2\left(\frac{N}{8}\right)^2 + \frac{N}{4} \\
 & \downarrow & \\
8\ \frac{N}{8} \text{ POINT DFT'S} + & 8\left(\frac{N}{8}\right)^2 + N + N + N & \left(\frac{N}{8}\right)^2 \longrightarrow 2\left(\frac{N}{16}\right)^2 + \frac{N}{8} \\
 & \vdots & \\
 & \downarrow & \\
 & \underbrace{N + N + \cdots + N}_{\nu \text{ TIMES}} &
\end{array}
$$

Class Review

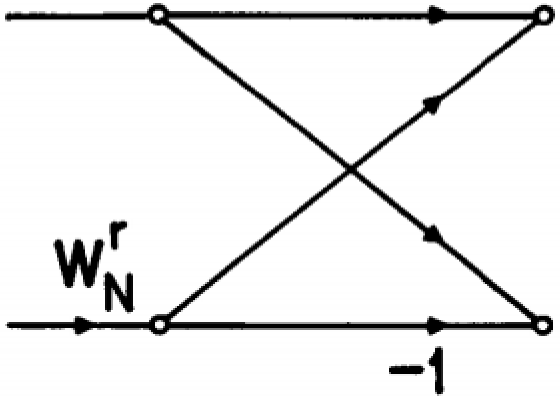


Basic butterfly computation in flow-graph j.

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

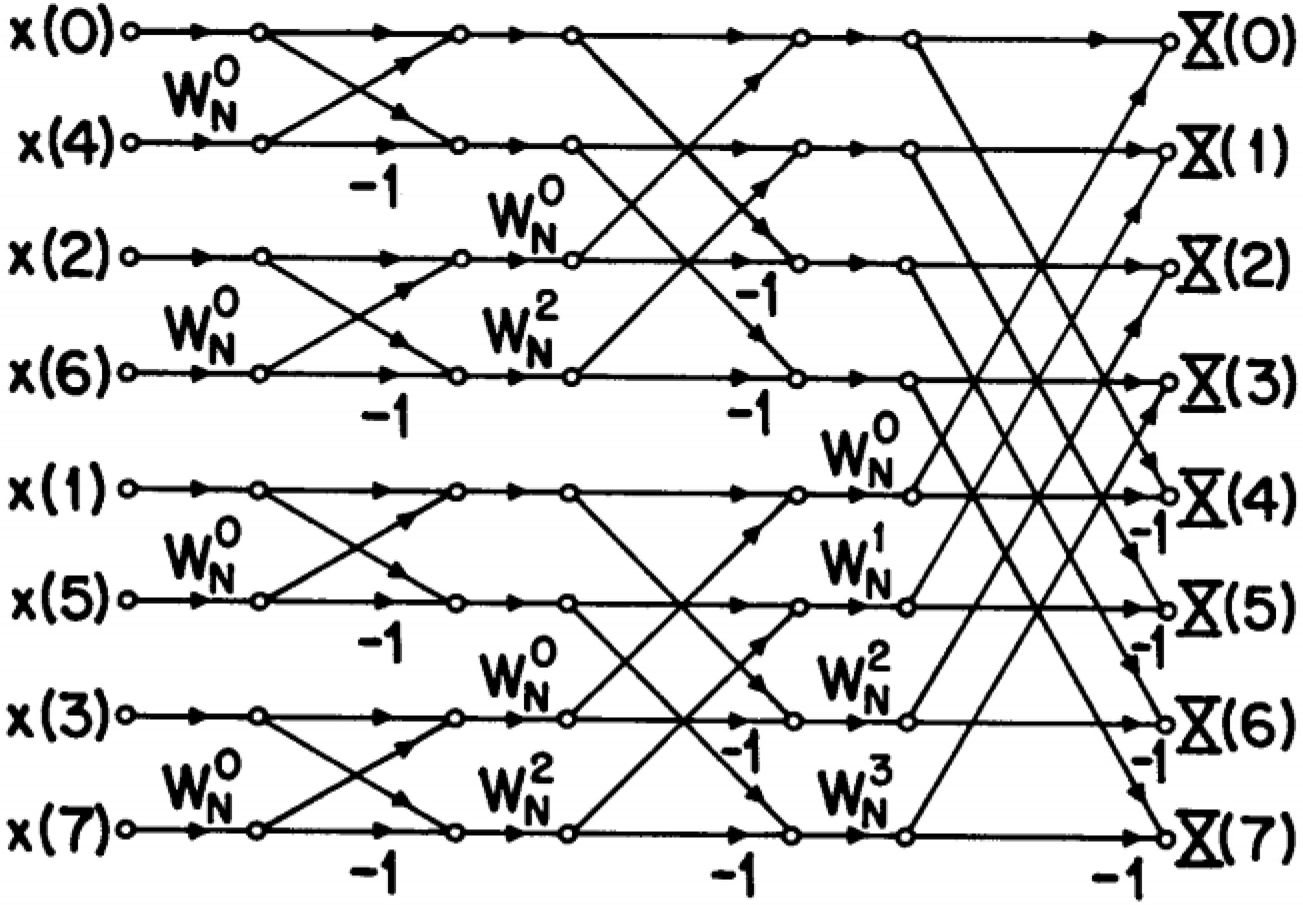
$$W_N^{\frac{N}{2}} = e^{-j\frac{2\pi}{N}\frac{N}{2}} = e^{-j\pi} = -1$$

Class Review



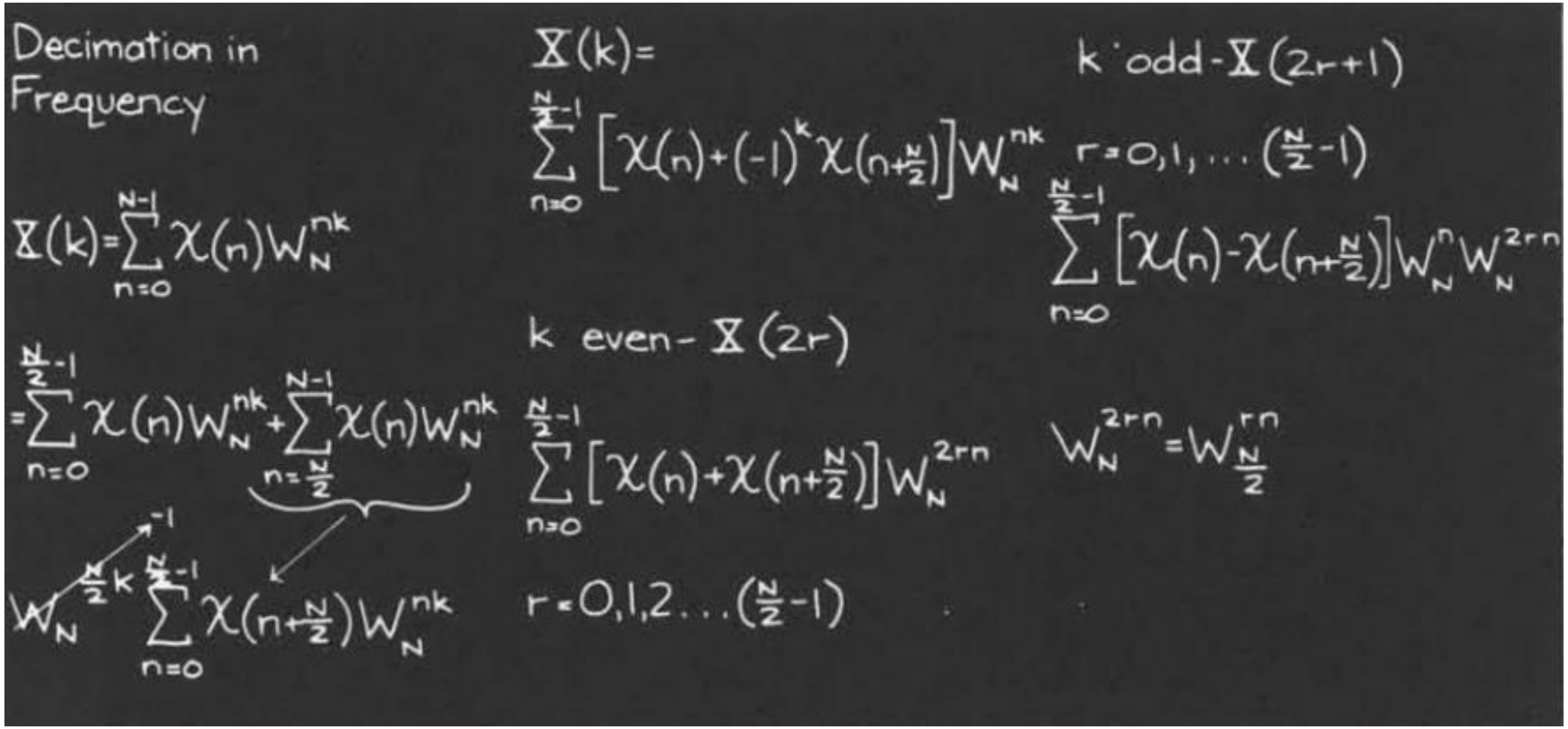
Rearrangement of the butterfly computation in 1.

Class Review



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

Class Review



Decimation in Frequency

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

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

$$\overset{-1}{W_N^{\frac{N}{2}k}}\sum_{n=0}^{\frac{N}{2}-1}x(n+\tfrac{N}{2})W_N^{nk}$$

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}\left[x(n)+(-1)^k x(n+\tfrac{N}{2})\right]W_N^{nk}$$

k even- $X(2r)$

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

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

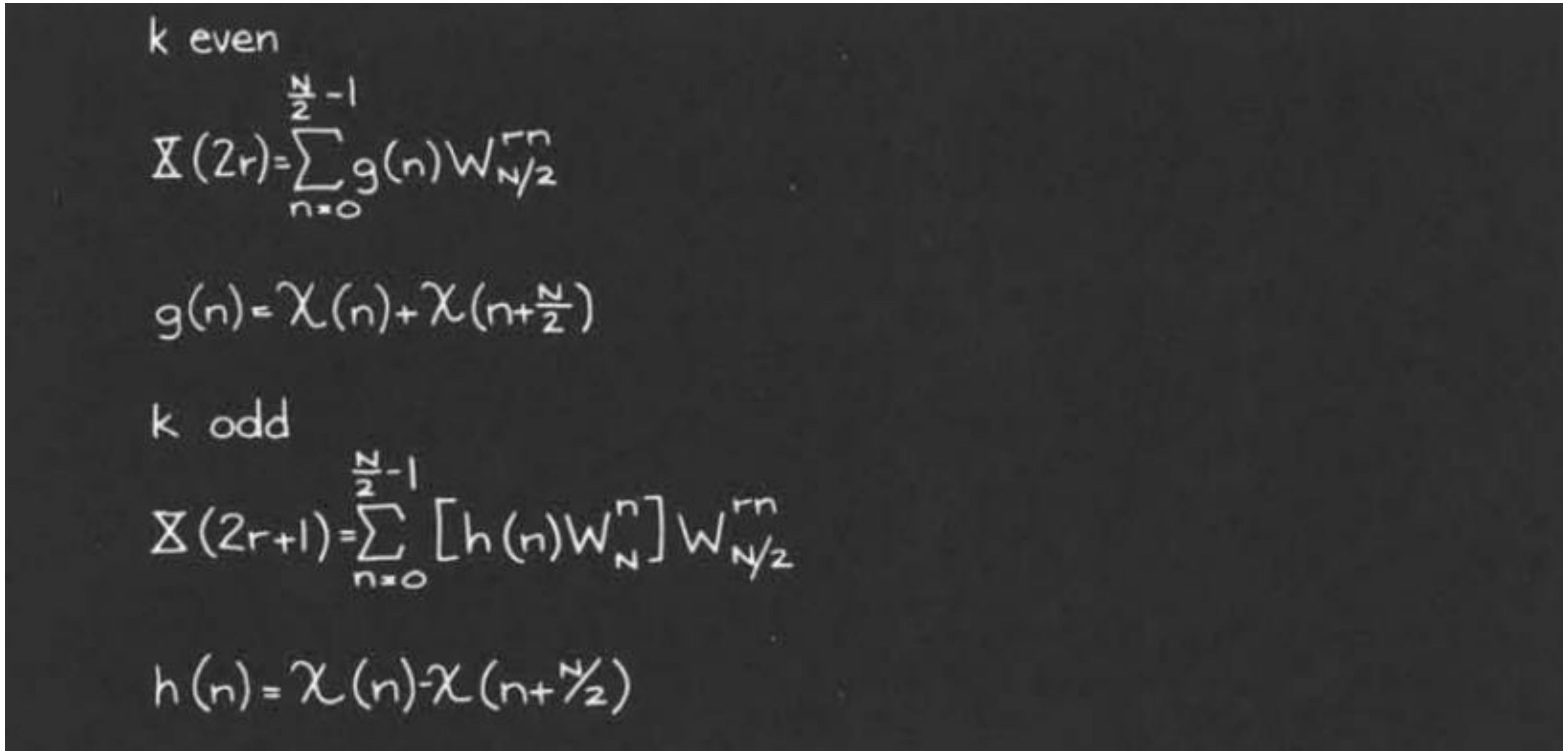
k odd- $X(2r+1)$

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

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

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

Class Review



k even

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

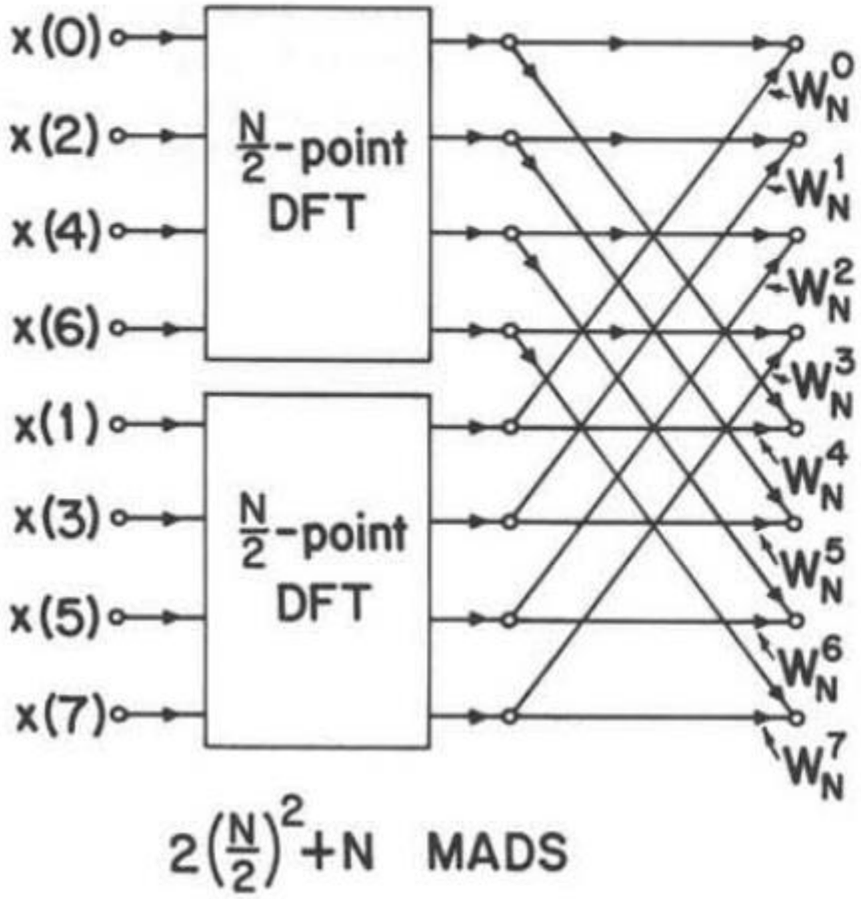
$$g(n)=x(n)+x(n+\tfrac{N}{2})$$

k odd

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

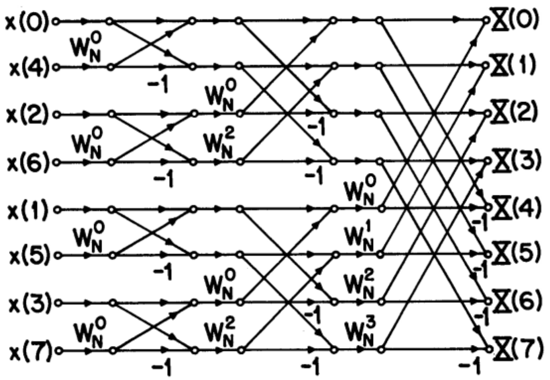
$$h(n)=x(n)-x(n+N/2)$$

Class Review



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

Class Review



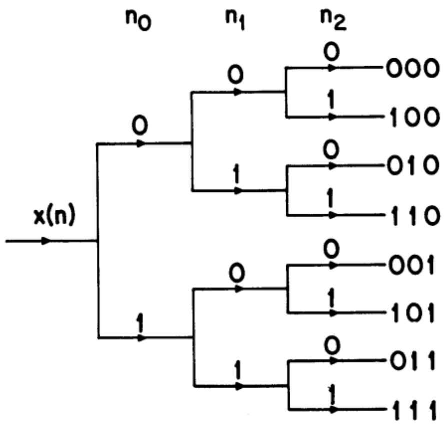
OPPENHEIM & SCHAFER 6-10

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

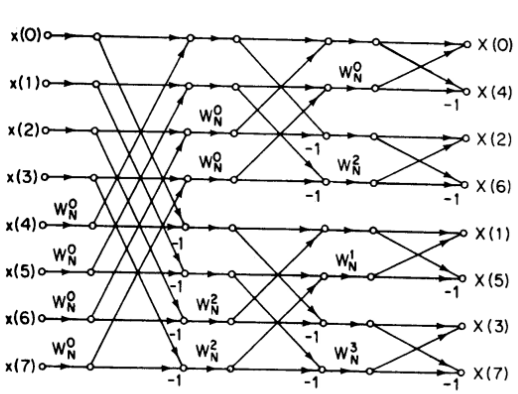
Relation between data index and data storage register.

$$x(n) = x\left(\cdots 2^2 n_2 + 2^1 n_1 + 2^0 n_0\right)$$



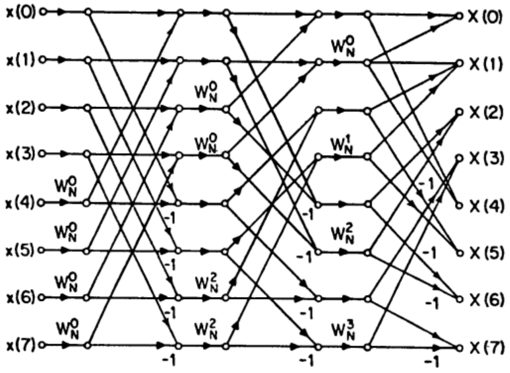
Sorting of data implied by the development of the decimation-in-time algorithm.

Class Review



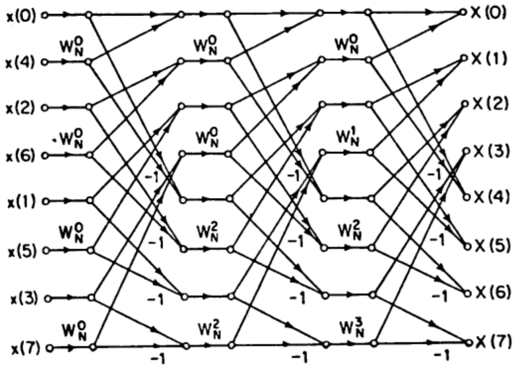
OPPENHEIM & SCHAFER 6-12

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



OPPENHEIM & SCHAFER 6-13

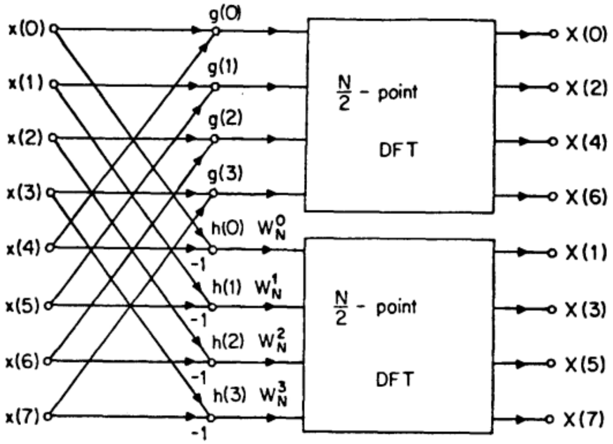
Rearrangement of flowgraph d. with both input and output in normal order.



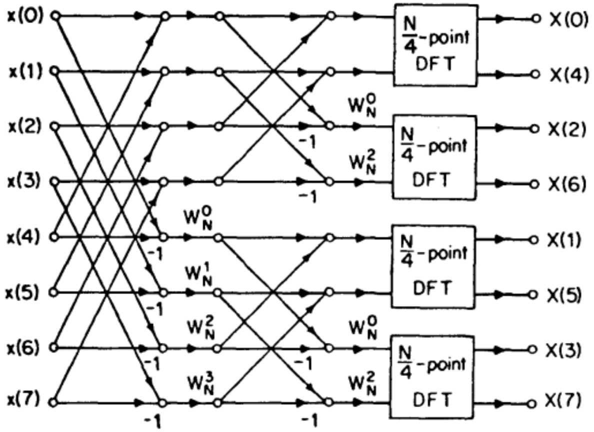
OPPENHEIM & SCHAFER 6-14

Rearrangement of flowgraph d having the same geometry for each stage.

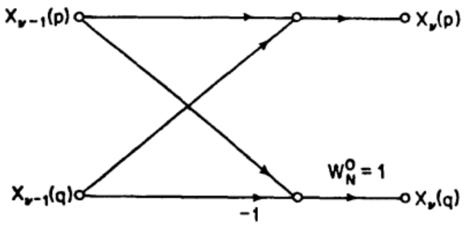
Class Review



Computation of even and odd-numbered DFT values.

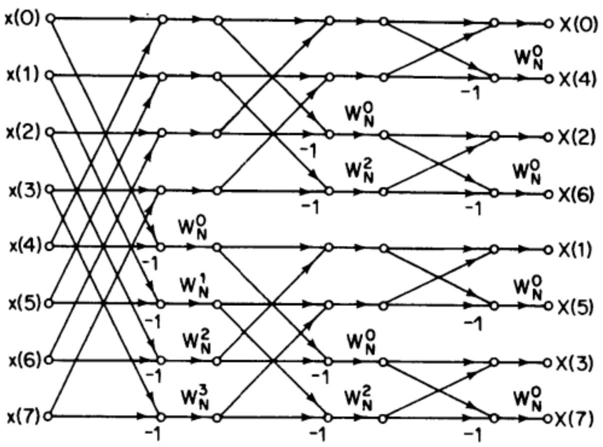


Decomposition of the N/2-point DFT's of flow-graph j.



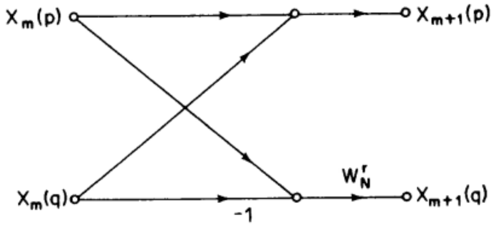
Flow-graph for two-point DFT.

Class Review

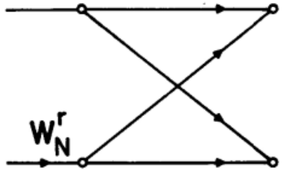


OPPENHEIM & SCHAFER 6-18

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

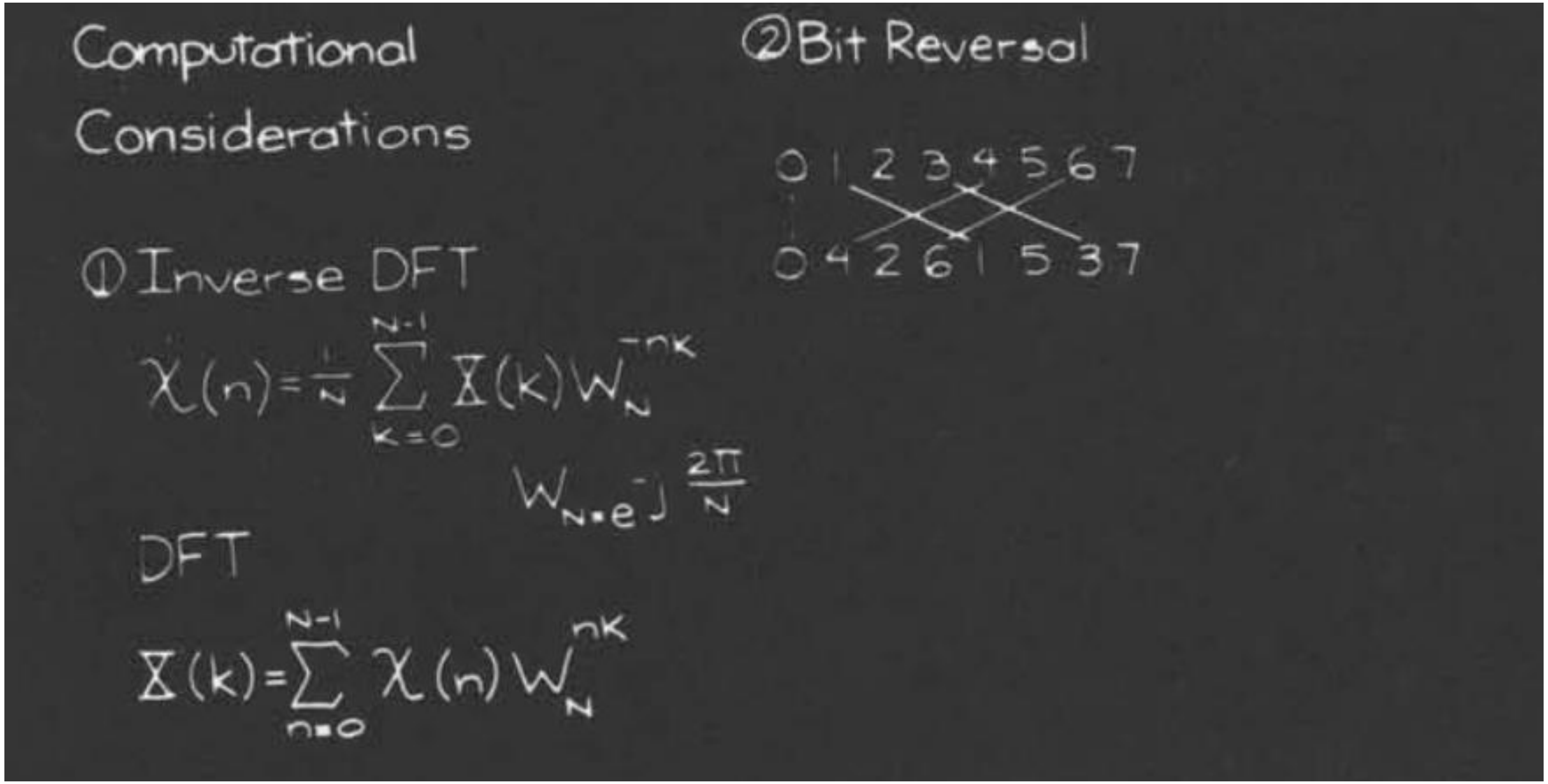


Flow-graph of a typical butterfly computation required in decimation-in-frequency FFT algorithm.



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

Class Review



Computational Considerations

① Inverse DFT

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

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

DFT

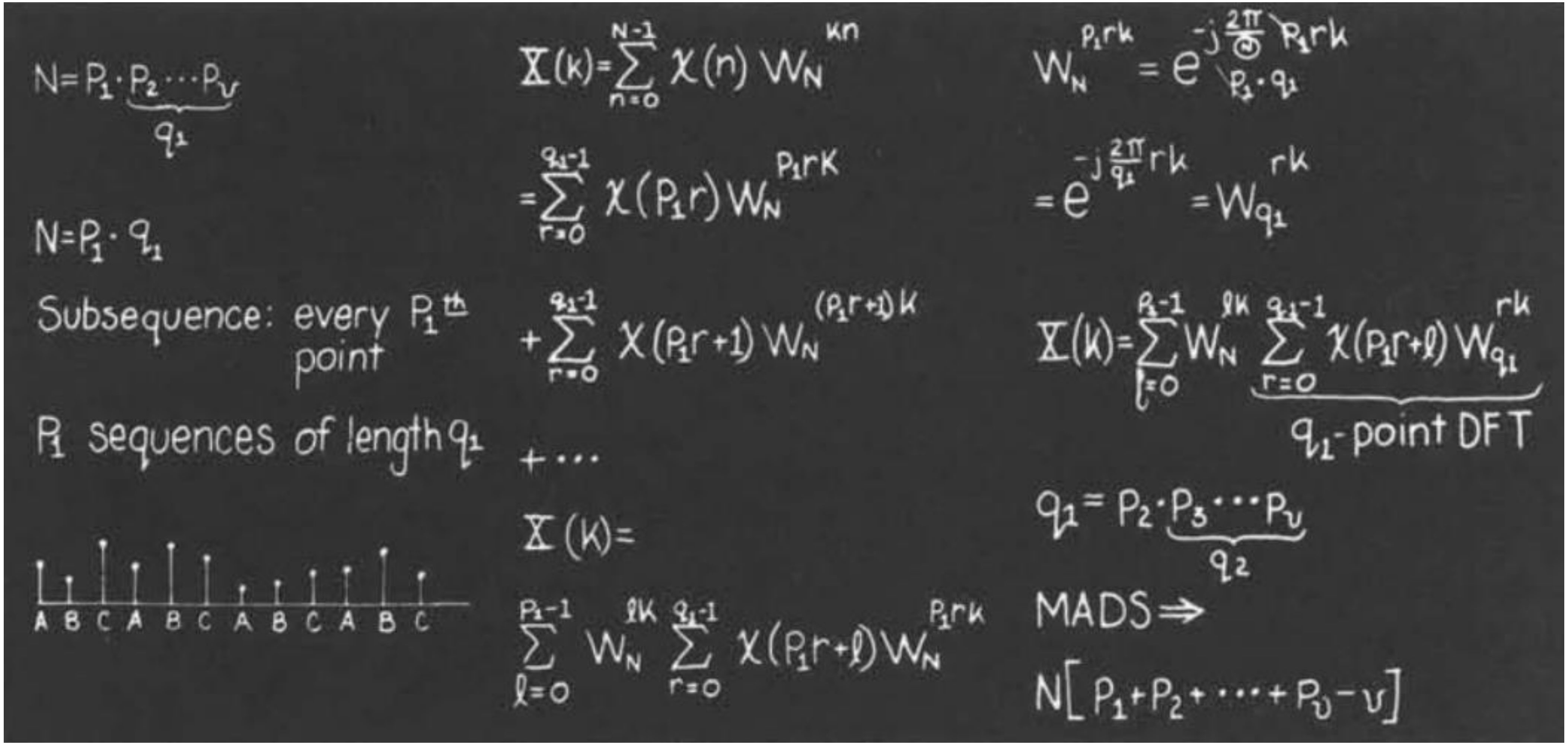
$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

② Bit Reversal

0 1 2 3 4 5 6 7

0 4 2 6 1 5 3 7

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

A B C A B C A B C A B C

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

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

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

$$+ \cdots$$

$$X(K) =$$

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

$$W_N^{P_1 r k} = e^{-j\frac{2\pi}{\underset{P_1 \cdot q_1}{\textcircled{N}}} P_1 r k}$$

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

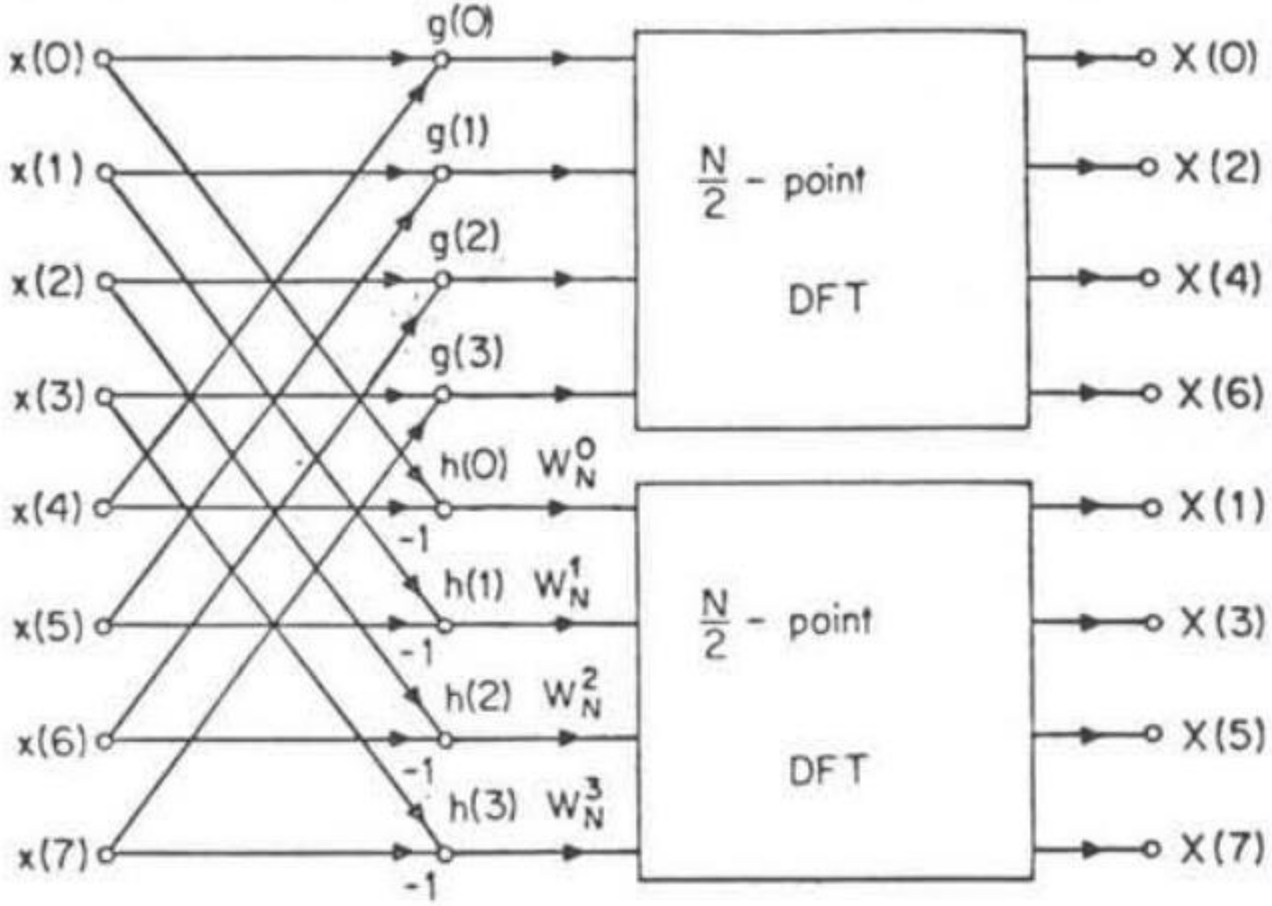
$$X(k) = \sum_{l=0}^{P_1-1} W_N^{lk} \underbrace{\sum_{r=0}^{q_1-1} x(P_1 r+l) W_{q_1}^{rk}}_{q_1\text{-point DFT}}$$

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

MADS ⇒

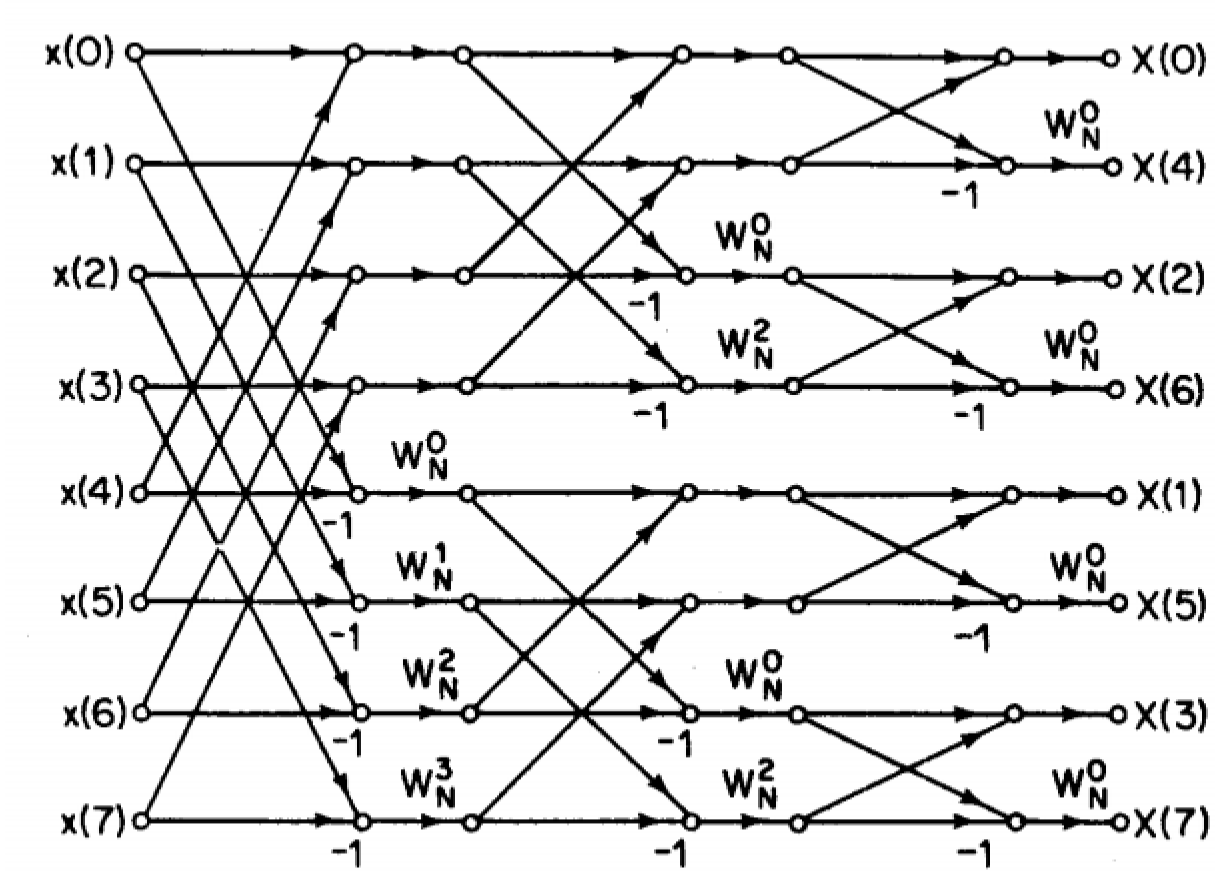
$N[P_1 + P_2 + \cdots + P_v - v]$

Class Review



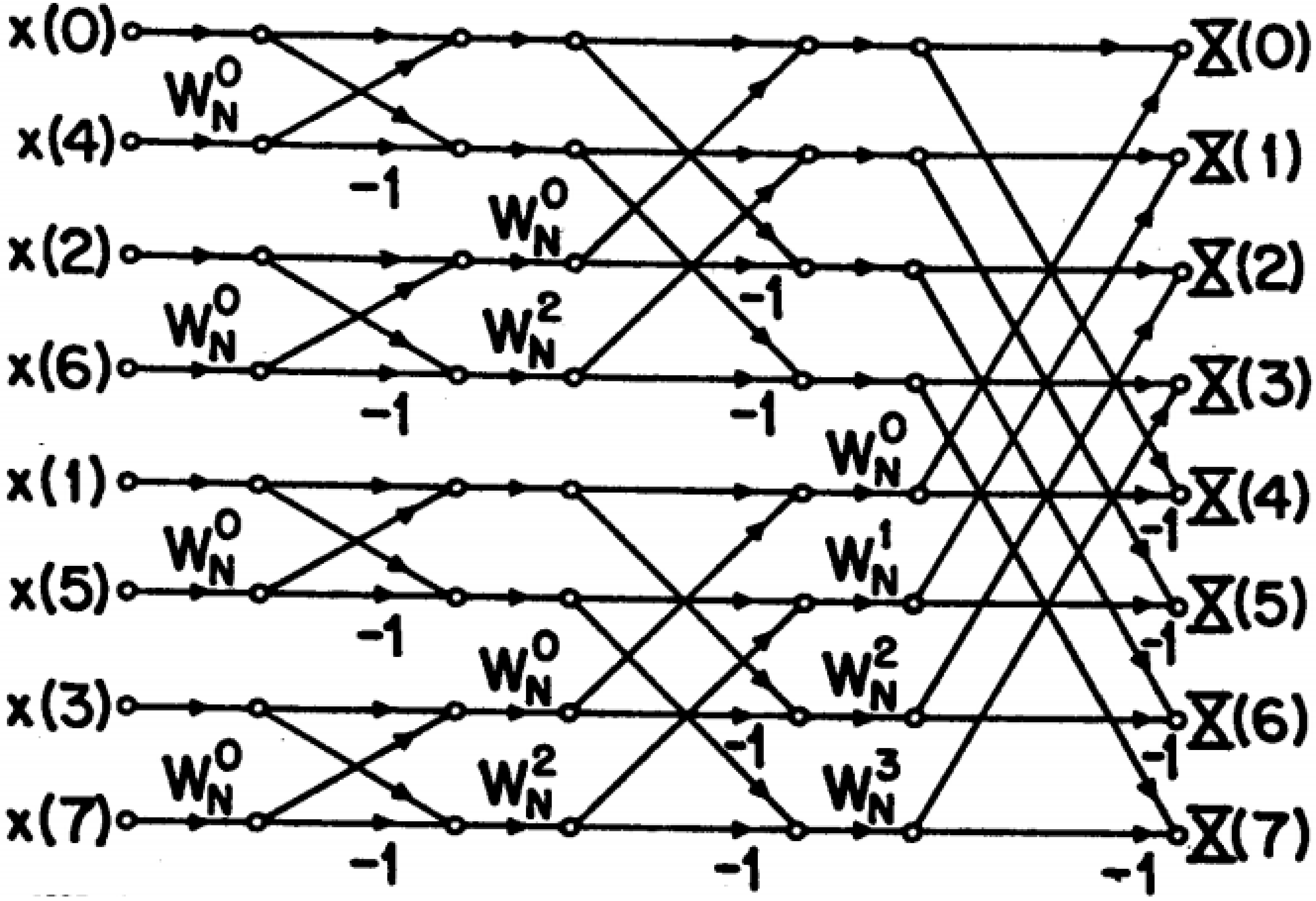
Computation of even and odd numbered DFT values.

Class Review



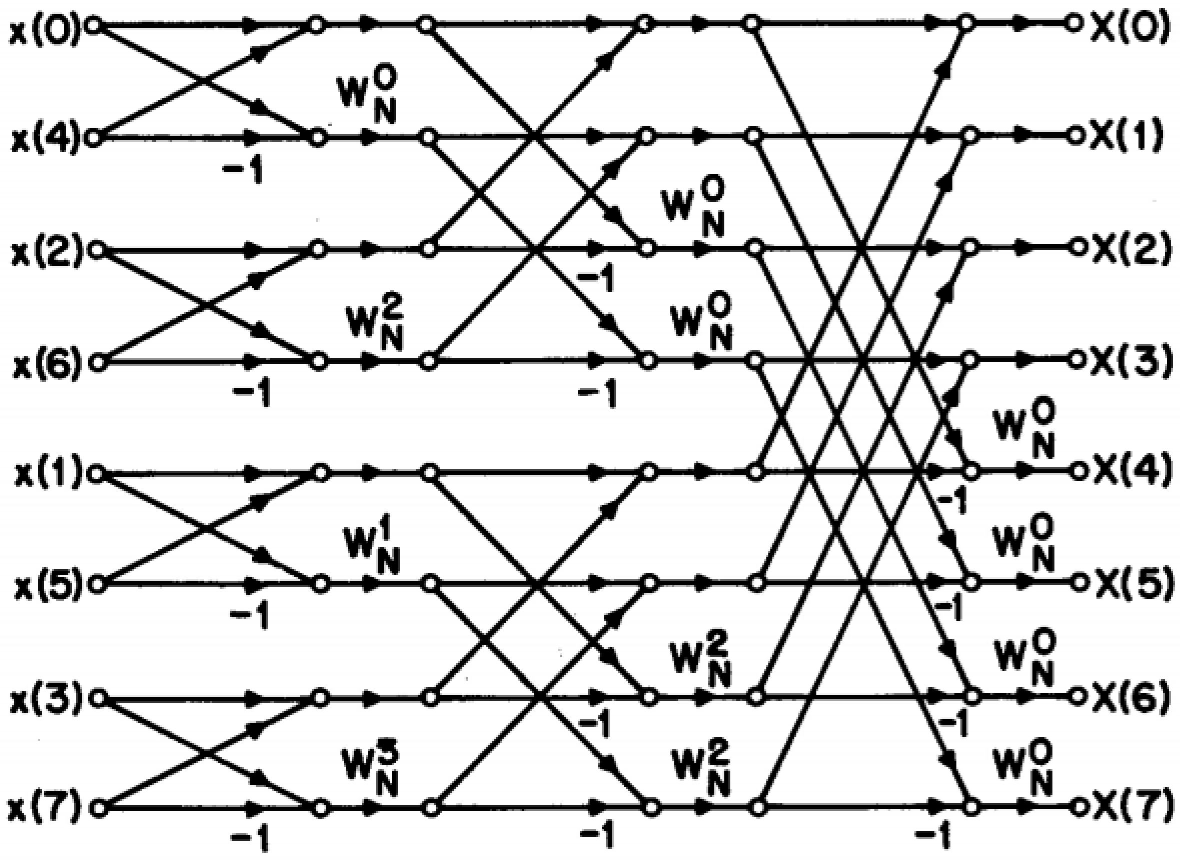
Flow-graph of complete decimation-in-frequency decomposition of an eight point DFT computation.

Class Review



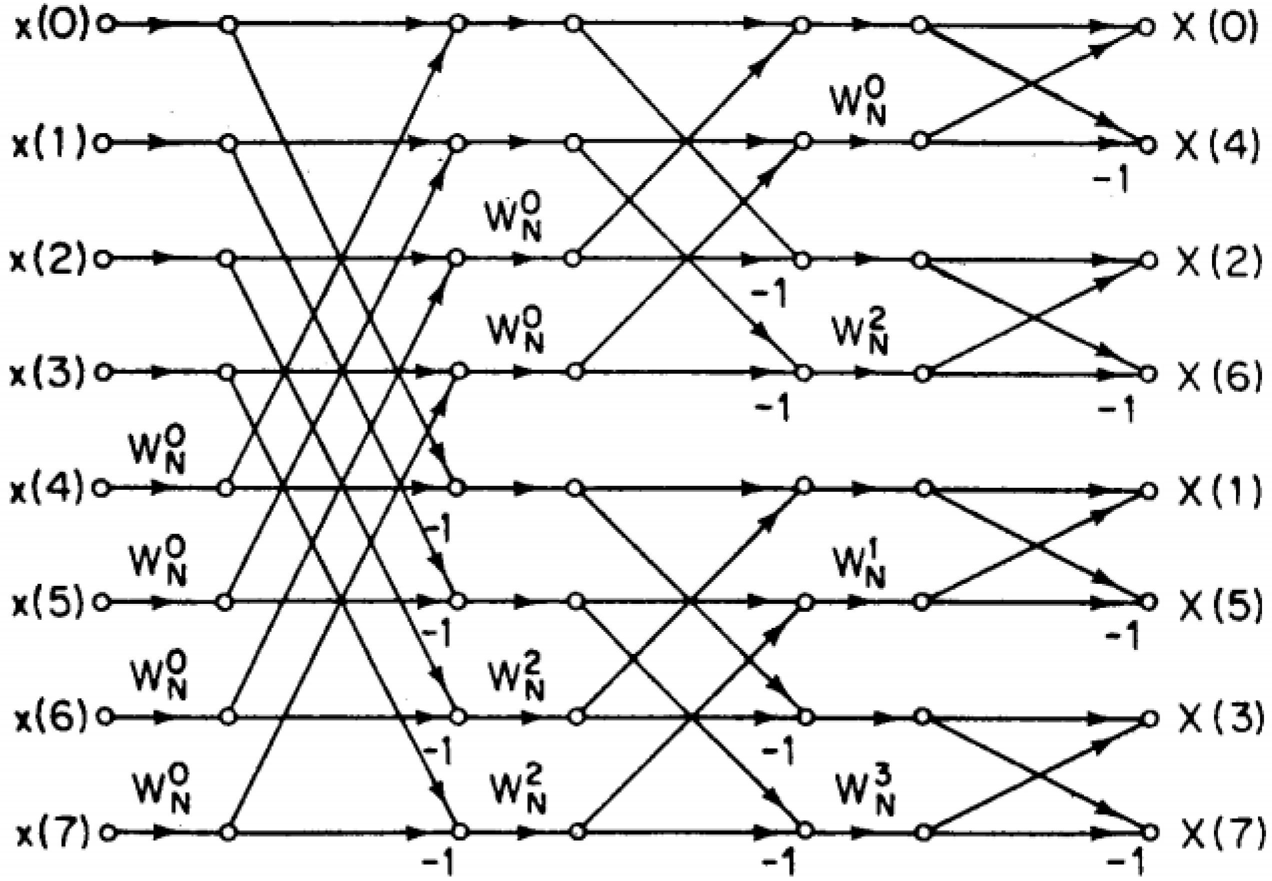
FFT flow-graph for decimation-in-time algorithm.

Class Review



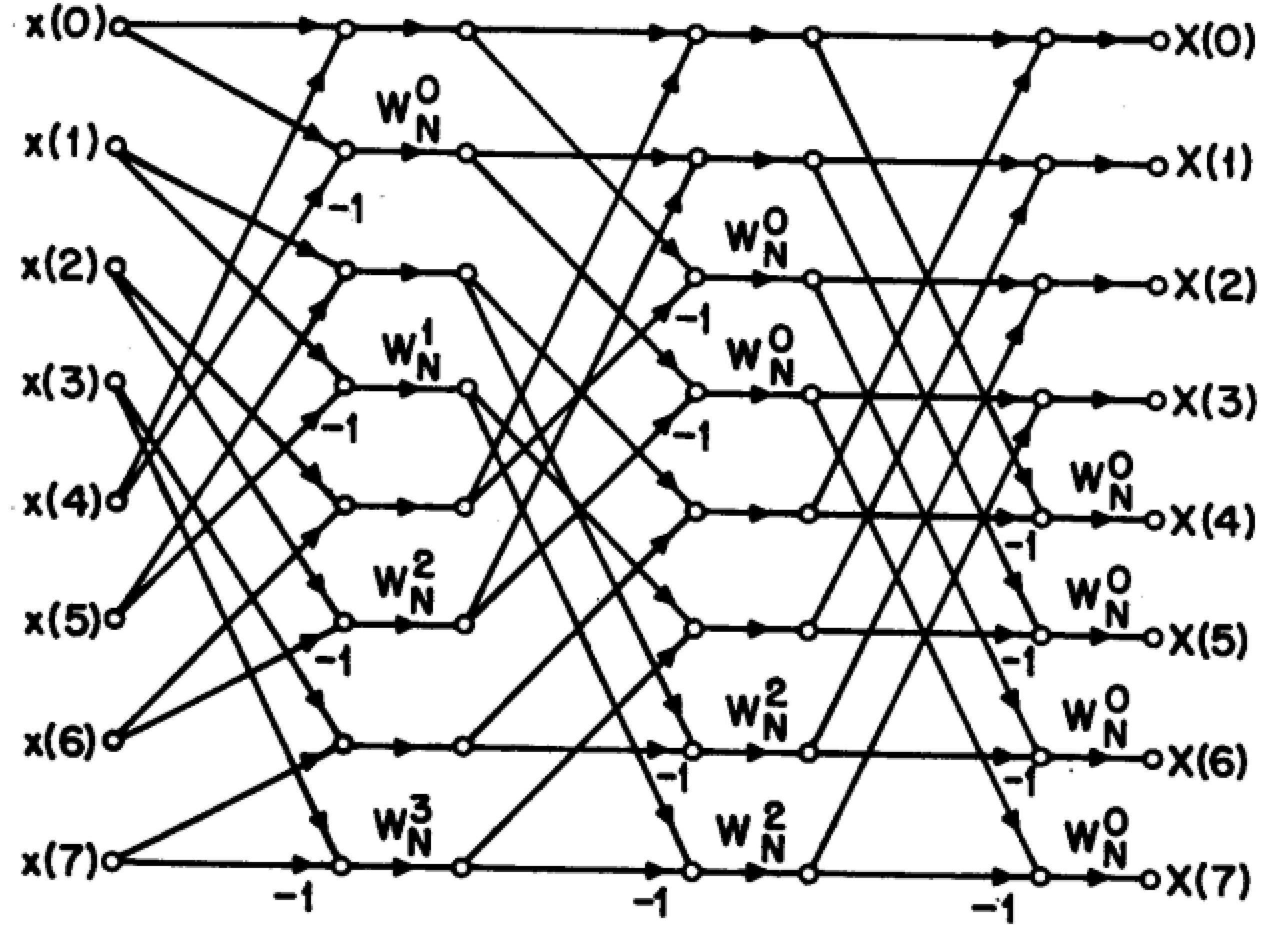
Rearrangement of the decimation-in-frequency flow-graph d. The input is now in bit-reversed order and the output is in normal order.

Class Review



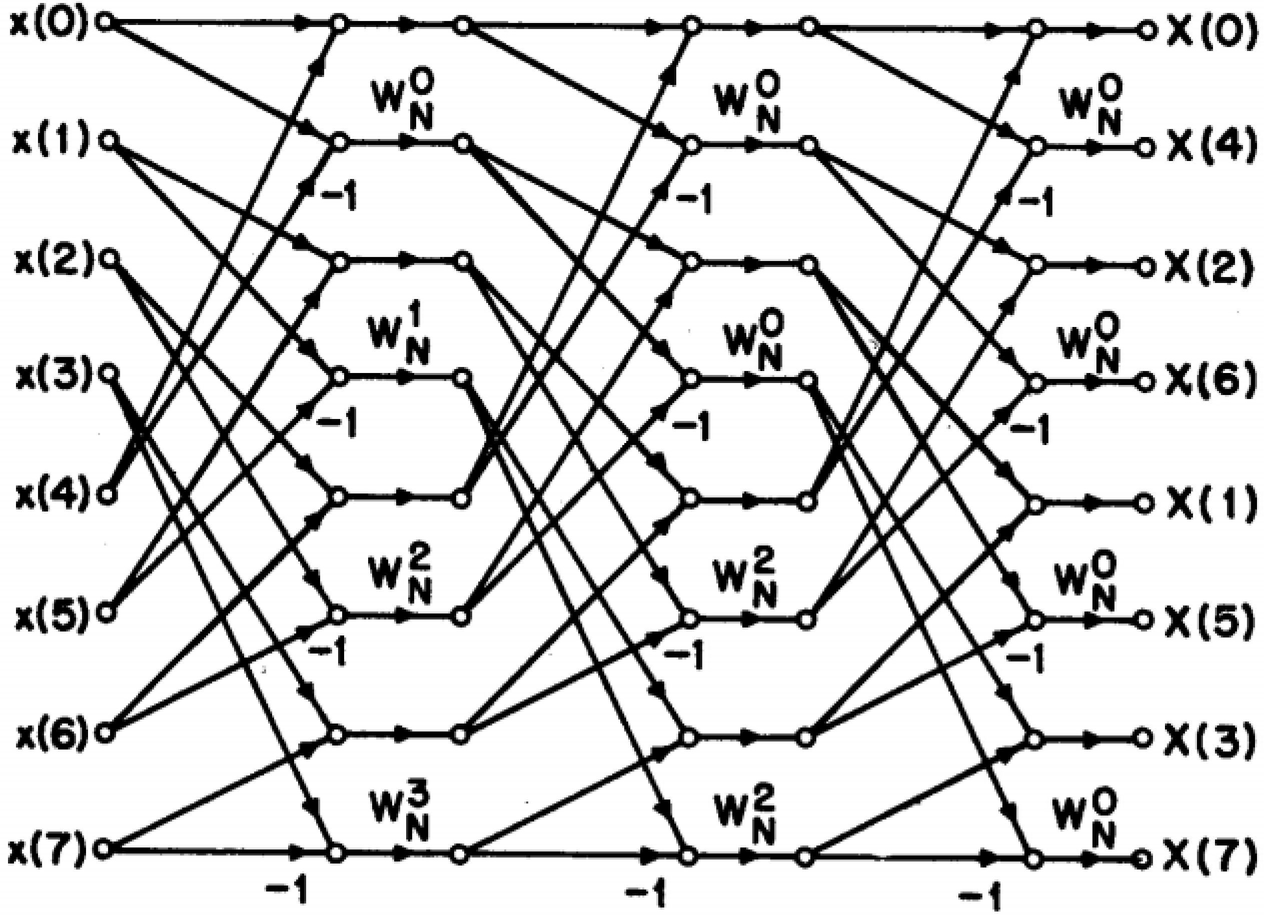
Rearrangement of e so that the input is in normal order and output in bit-reversed order.

Class Review



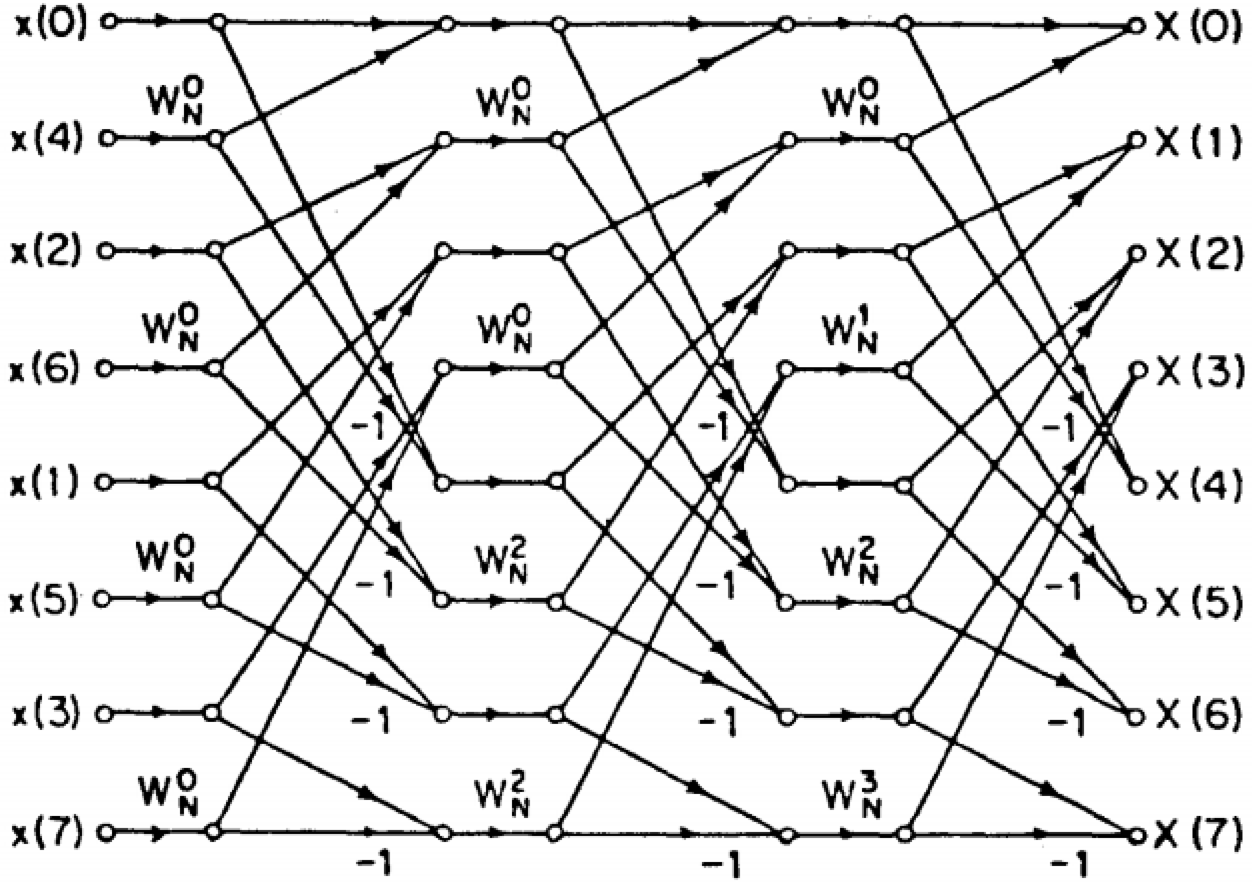
Rearrangement of d so that both input and output are in normal order.

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.

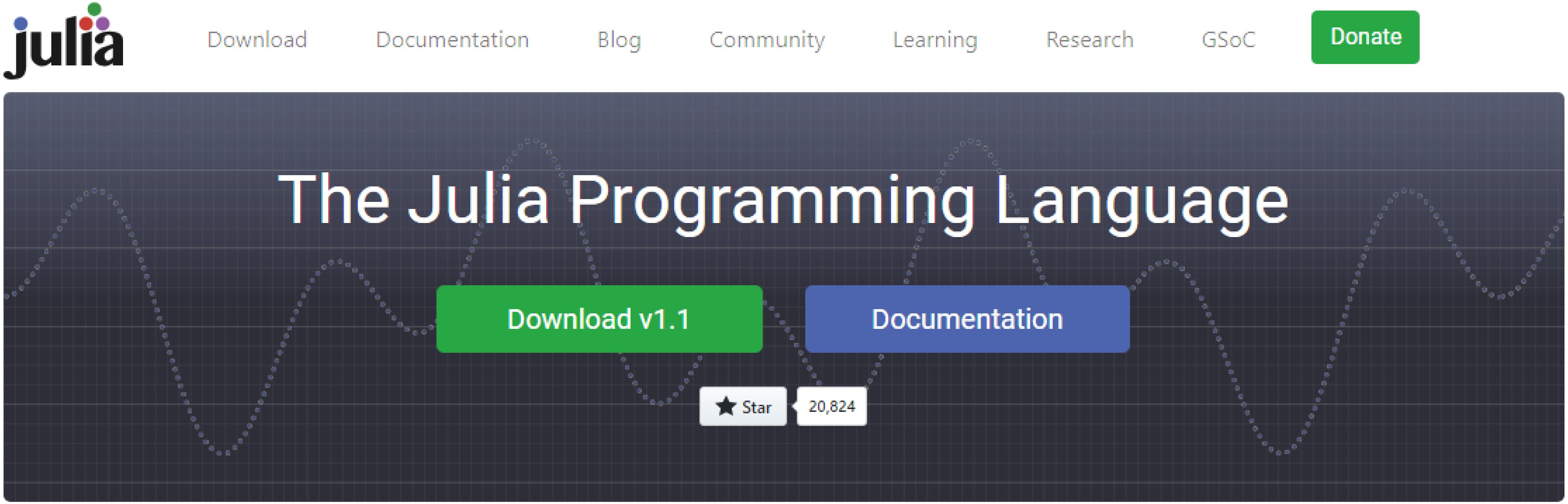
Optimization

Convex Optimization

Convex.jl , cvx, cvxpy, cvxr

Julia Programming Language for Optimization

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

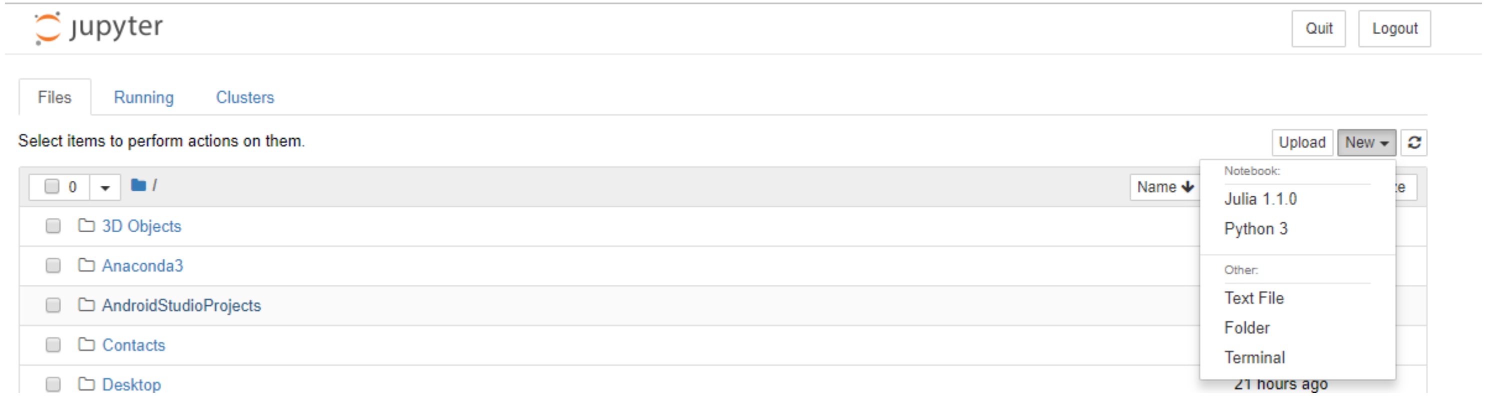


Julia Programming Language for Optimization

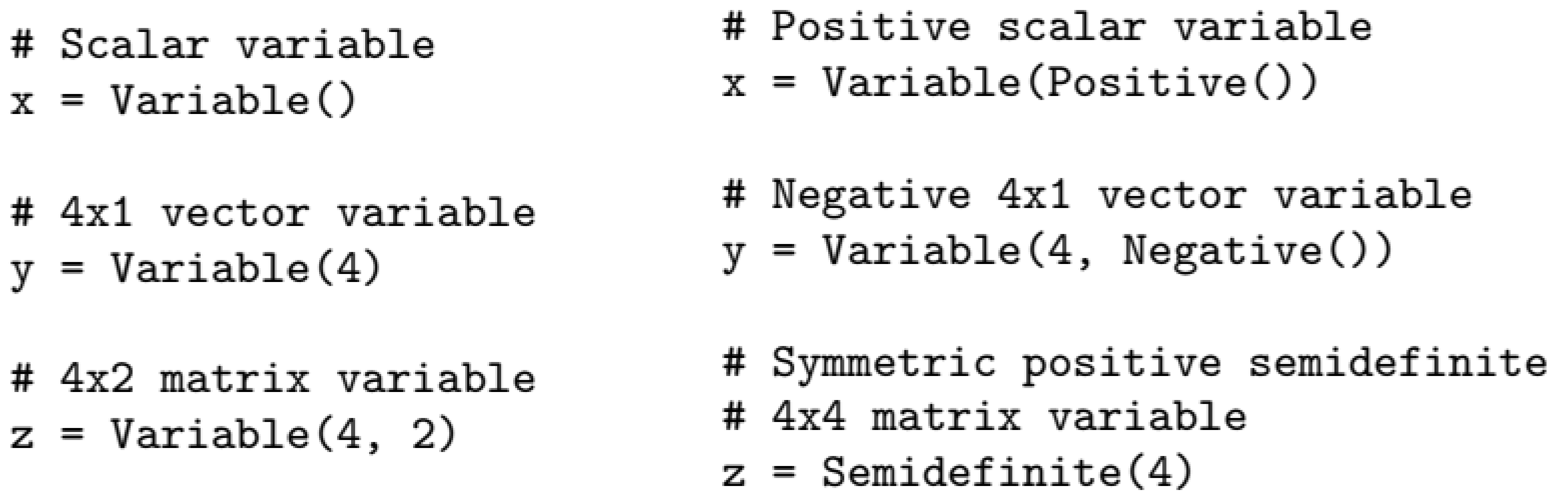
Import Pkg
Pkg.add(“IJulia”)

Import Pkg
Pkg.add(“SCS”)
Pkg.add(“Convex”)

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



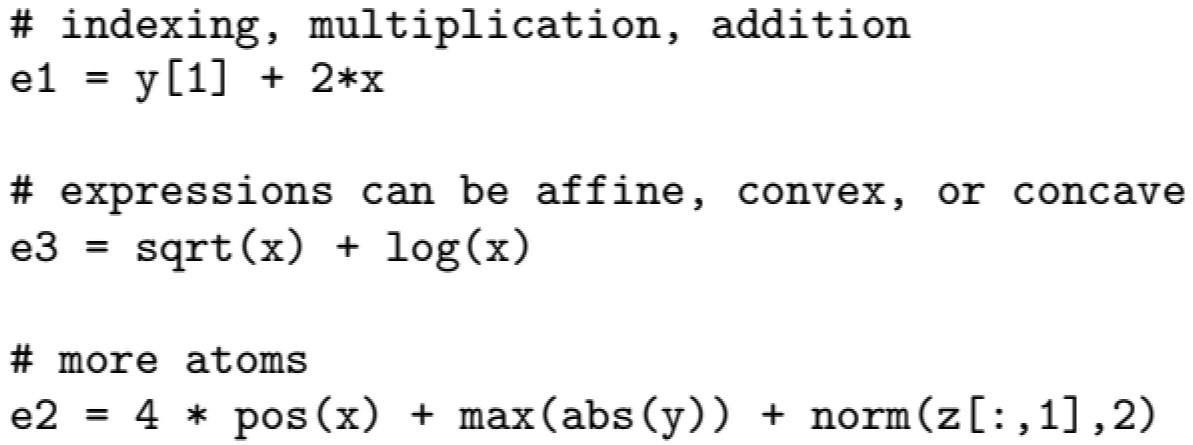
Select items to perform actions on them.



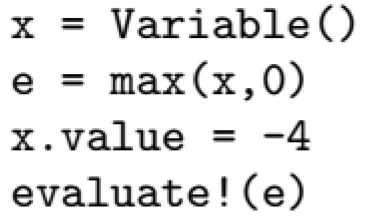
```
# Scalar variable
x = Variable()

# 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

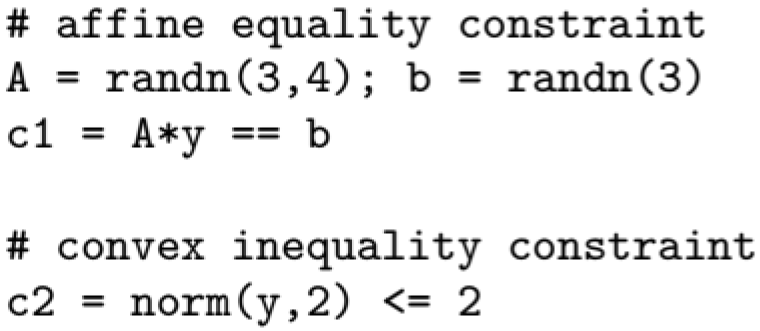


```
# indexing, multiplication, addition
e1 = y[1] + 2*x

# 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)
```

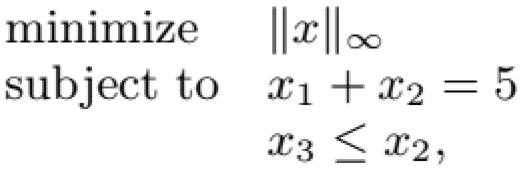


```
x = Variable()
e = max(x,0)
x.value = -4
evaluate!(e)
```

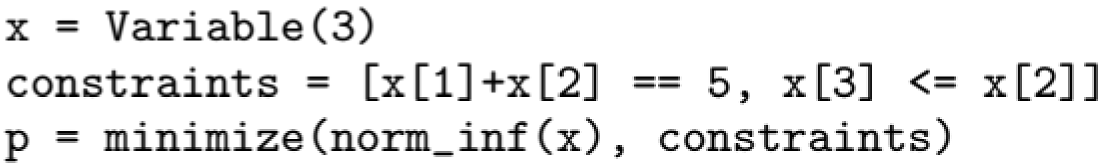
Julia Programming Language for Optimization



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

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

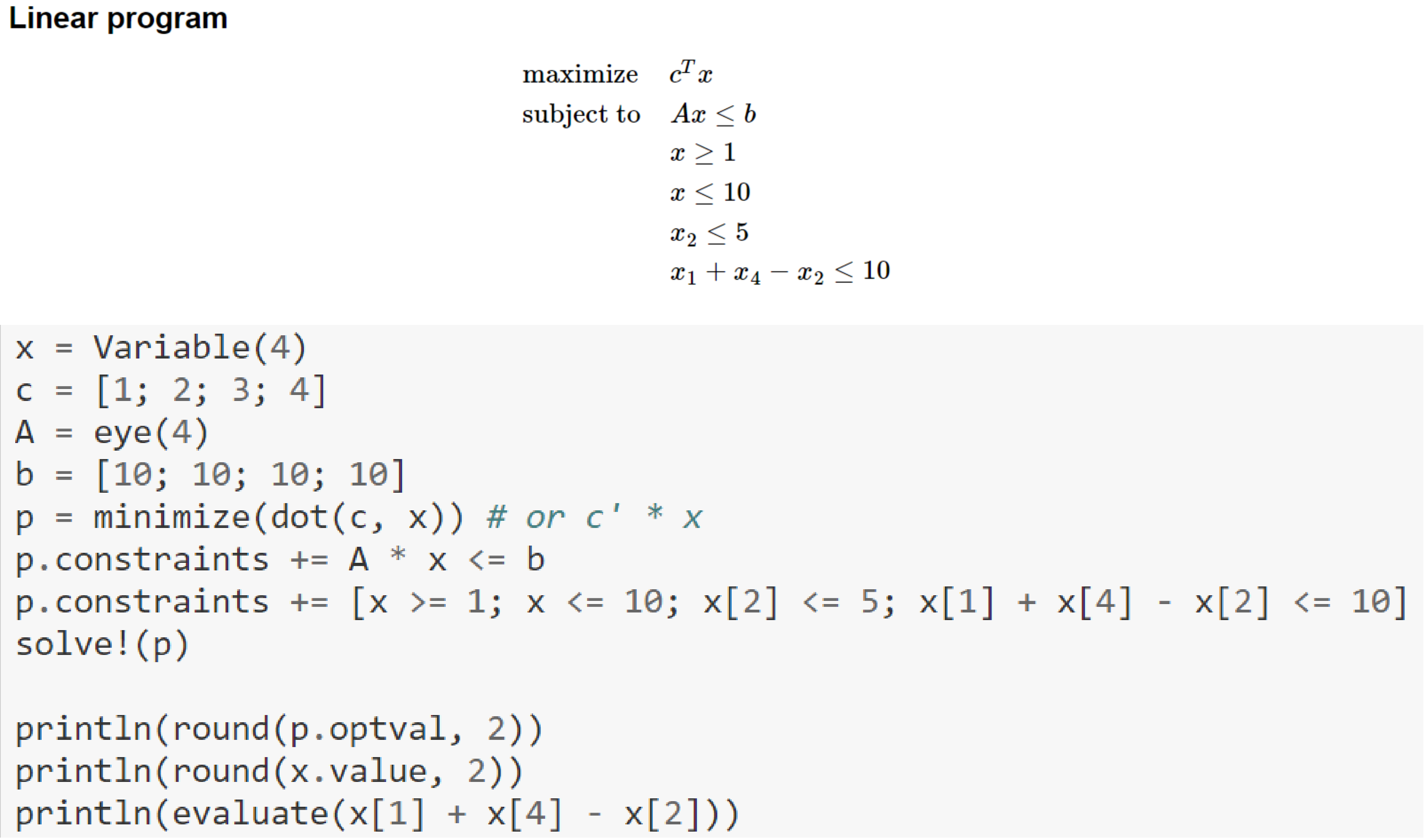


$$\begin{array}{ll} \text{minimize} & \|x\|_\infty \\ \text{subject to} & x_1 + x_2 = 5 \\ & x_3 \leq x_2, \end{array}$$



```
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

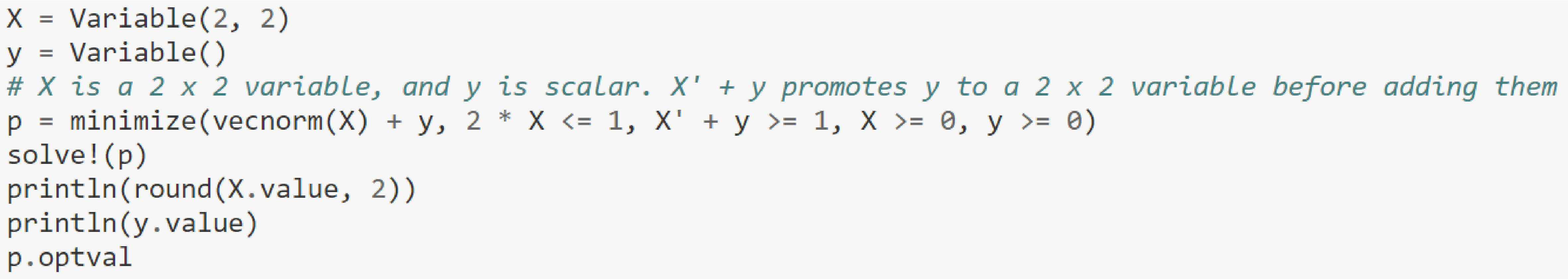
$$
\begin{array}{ll}
\text{maximize} & c^T x \\
\text{subject to} & Ax \leq b \\
 & x \geq 1 \\
 & x \leq 10 \\
 & x_2 \leq 5 \\
 & x_1 + x_4 - x_2 \leq 10
\end{array}
$$

```
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

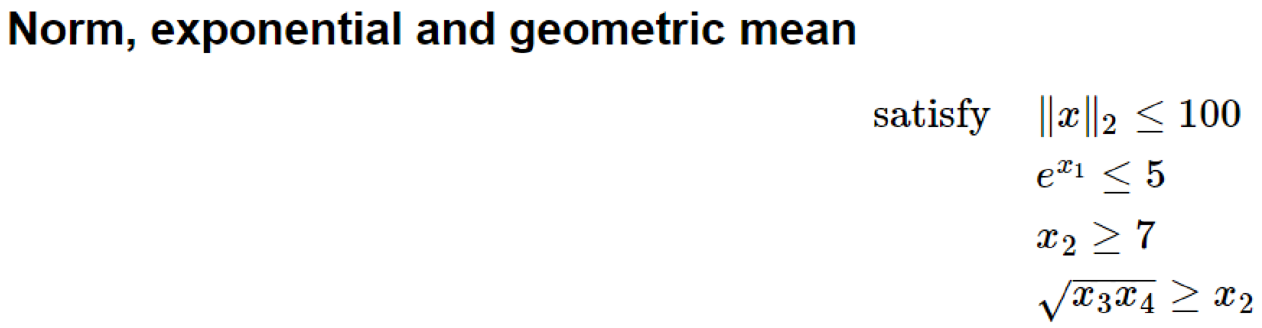
Matrix Variables and promotions

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



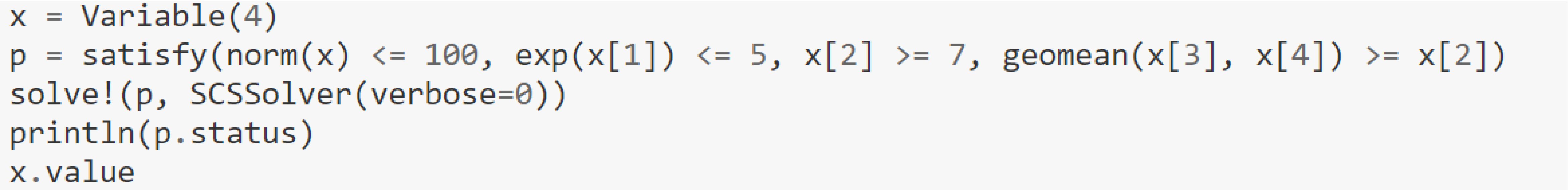
```
X = Variable(2, 2)
y = Variable()
# X is a 2 x 2 variable, and y is scalar. X' + y promotes y to a 2 x 2 variable before adding them
p = minimize(vecnorm(X) + y, 2 * X <= 1, X' + y >= 1, X >= 0, y >= 0)
solve!(p)
println(round(X.value, 2))
println(y.value)
p.optval
```

Julia Programming Language for Optimization



Norm, exponential and geometric mean

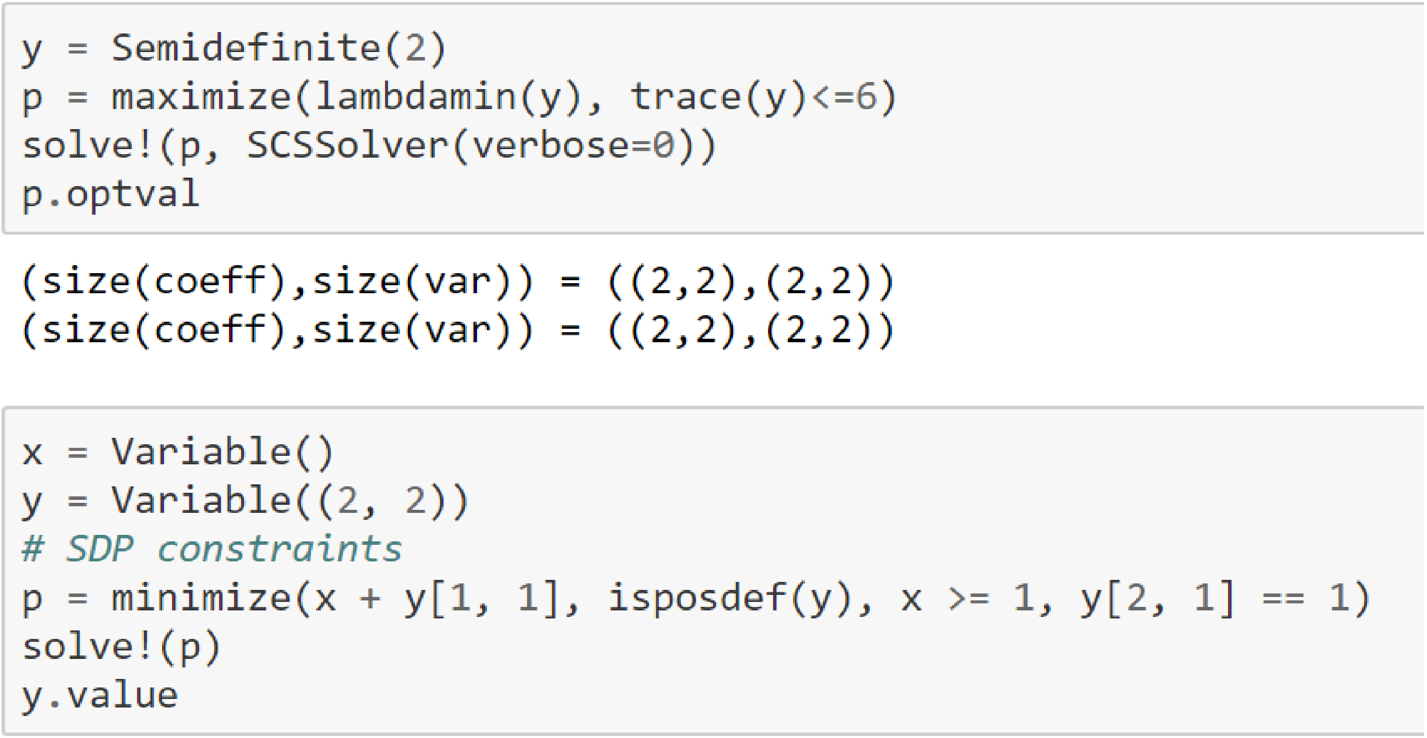
$$\begin{array}{ll} \text{satisfy} & \|x\|_2 \leq 100 \\ & e^{x_1} \leq 5 \\ & x_2 \geq 7 \\ & \sqrt{x_3 x_4} \geq x_2 \end{array}$$



```
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
```

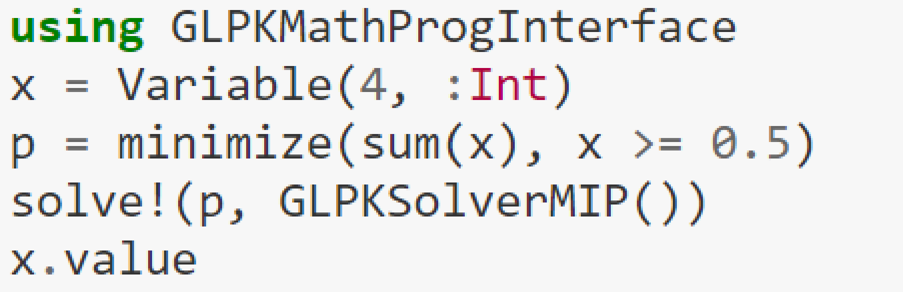
```
(size(coeff),size(var)) = ((2,2),(2,2))
(size(coeff),size(var)) = ((2,2),(2,2))
```

```
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

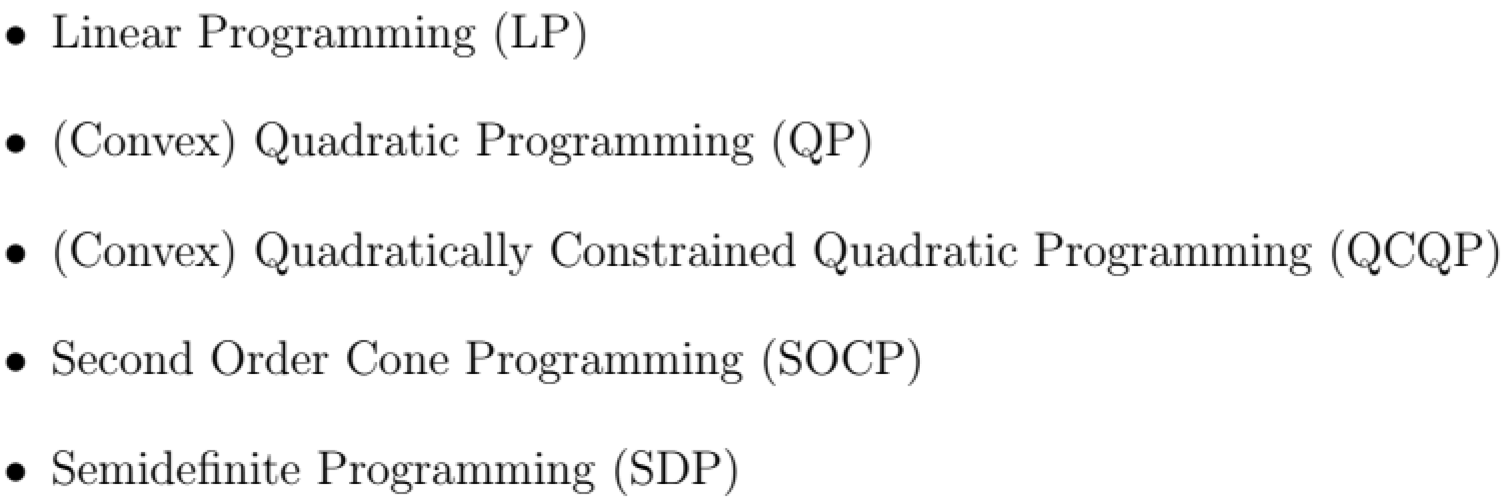
Mixed integer program

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



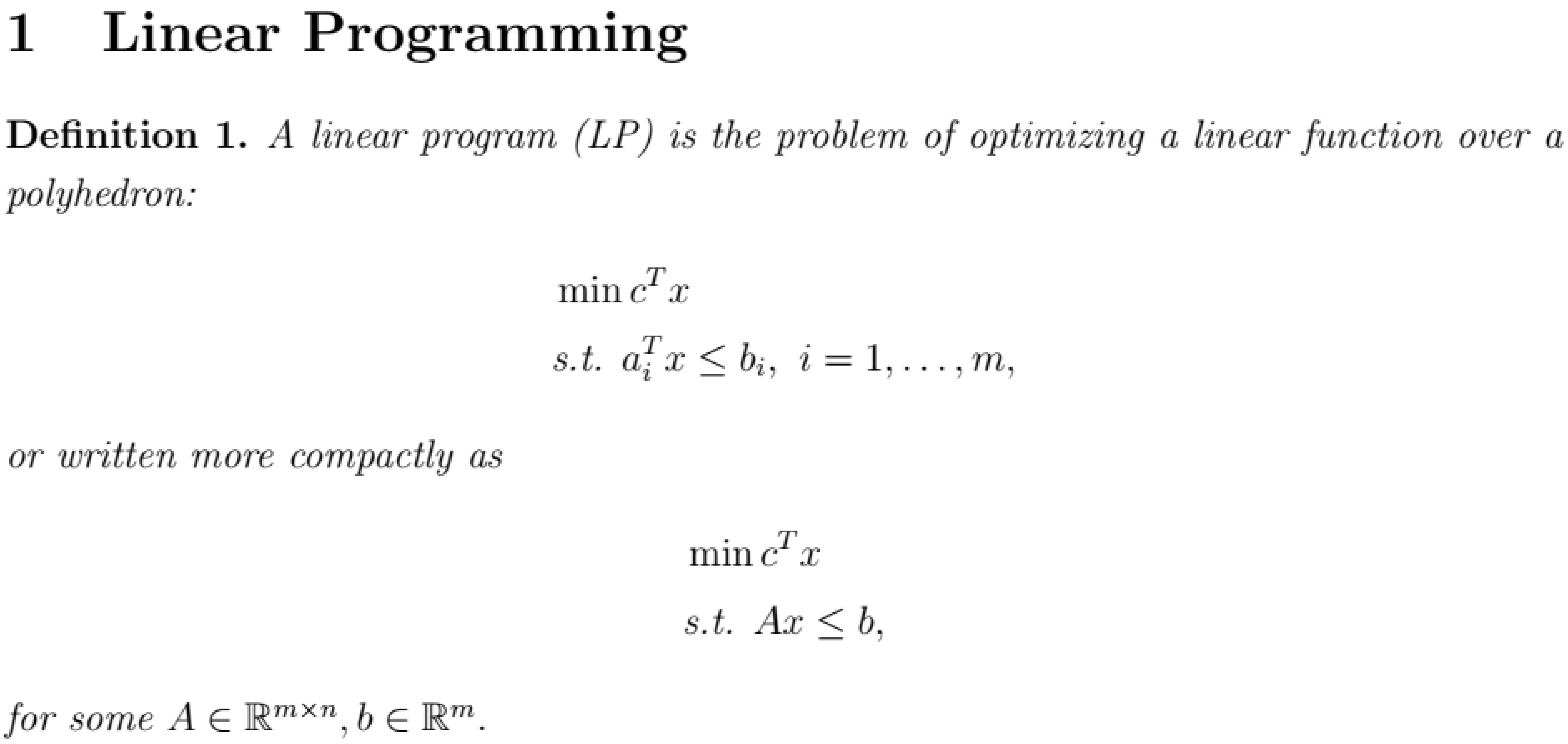
```
using GLPKMathProgInterface
x = Variable(4, :Int)
p = minimize(sum(x), x >= 0.5)
solve!(p, GLPKSolverMIP())
x.value
```

Julia Programming Language for Optimization



- Linear Programming (LP)
- (Convex) Quadratic Programming (QP)
- (Convex) Quadratically Constrained Quadratic Programming (QCQP)
- Second Order Cone Programming (SOCP)
- Semidefinite Programming (SDP)

Julia Programming Language for Optimization



1 Linear Programming

Definition 1. *A linear program (LP) is the problem of optimizing a linear function over a polyhedron:*

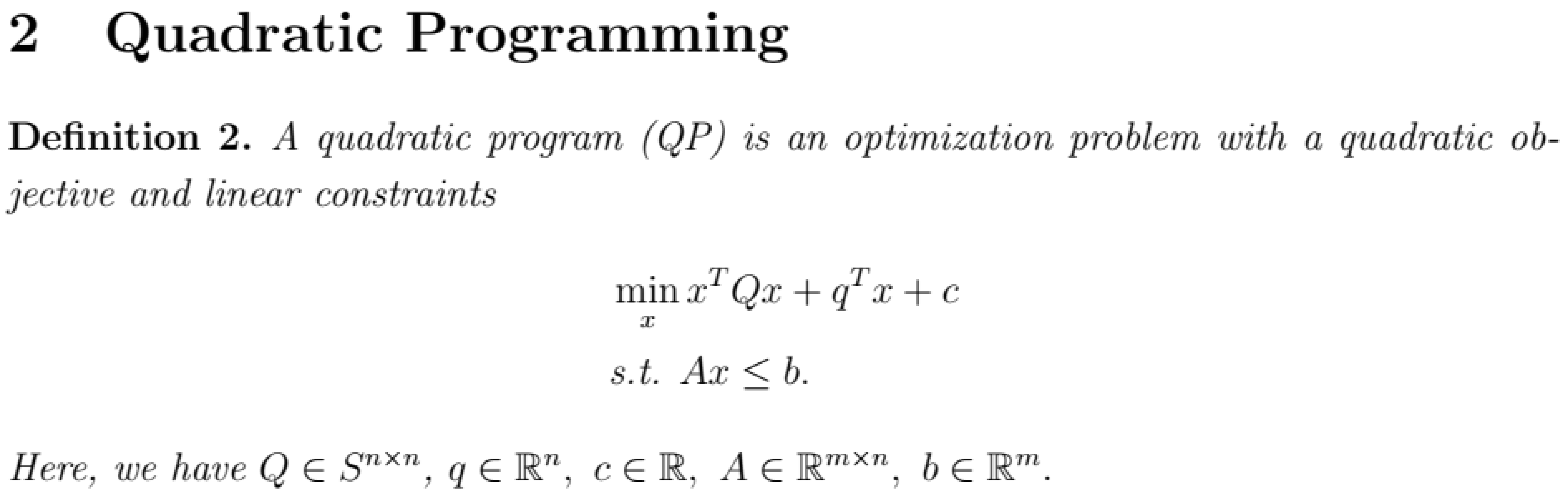
$$
\begin{aligned}
&\min c^T x \\
&s.t.\ \ a_i^T x \le b_i,\ \ i = 1, \dots, m,
\end{aligned}
$$

or written more compactly as

$$
\begin{aligned}
&\min c^T x \\
&s.t.\ \ Ax \le b,
\end{aligned}
$$

for some $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$.

Julia Programming Language for Optimization



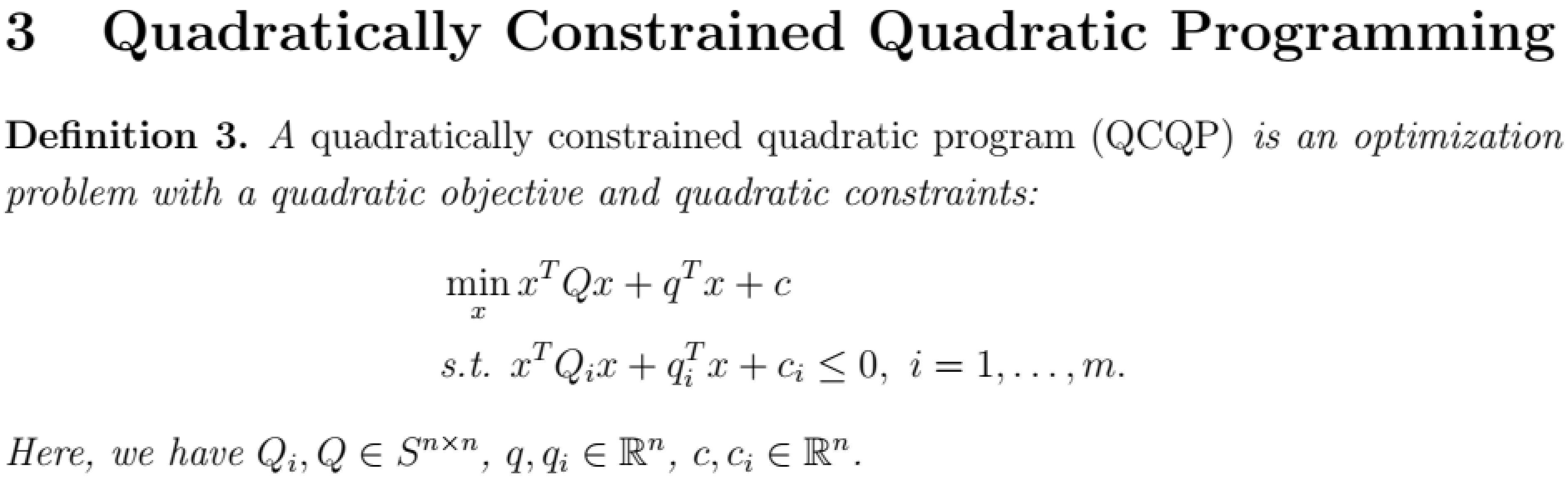
2 Quadratic Programming

Definition 2. *A quadratic program (QP) is an optimization problem with a quadratic objective and linear constraints*

$$\begin{aligned} &\min_{x} x^T Q x + q^T x + c \\ &s.t.\ \ Ax \leq b. \end{aligned}$$

Here, we have $Q \in S^{n\times n},\ q \in \mathbb{R}^n,\ c \in \mathbb{R},\ A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m.$

Julia Programming Language for Optimization



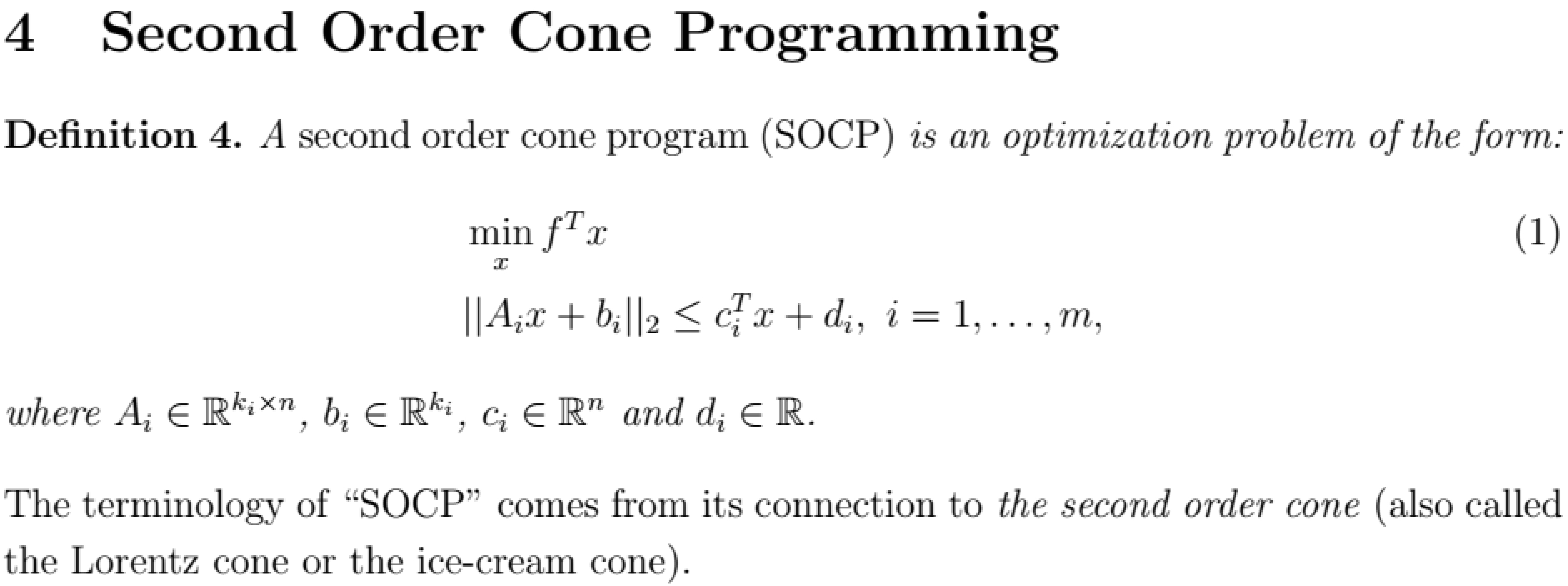
3 Quadratically Constrained Quadratic Programming

Definition 3. *A* quadratically constrained quadratic program (QCQP) *is an optimization problem with a quadratic objective and quadratic constraints:*

$$\min_x x^T Q x + q^T x + c$$
$$s.t.\ \ x^T Q_i x + q_i^T x + c_i \leq 0,\ \ i = 1, \ldots, m.$$

Here, we have $Q_i, Q \in S^{n \times n}$, $q, q_i \in \mathbb{R}^n$, $c, c_i \in \mathbb{R}^n$.

Julia Programming Language for Optimization



4 Second Order Cone Programming

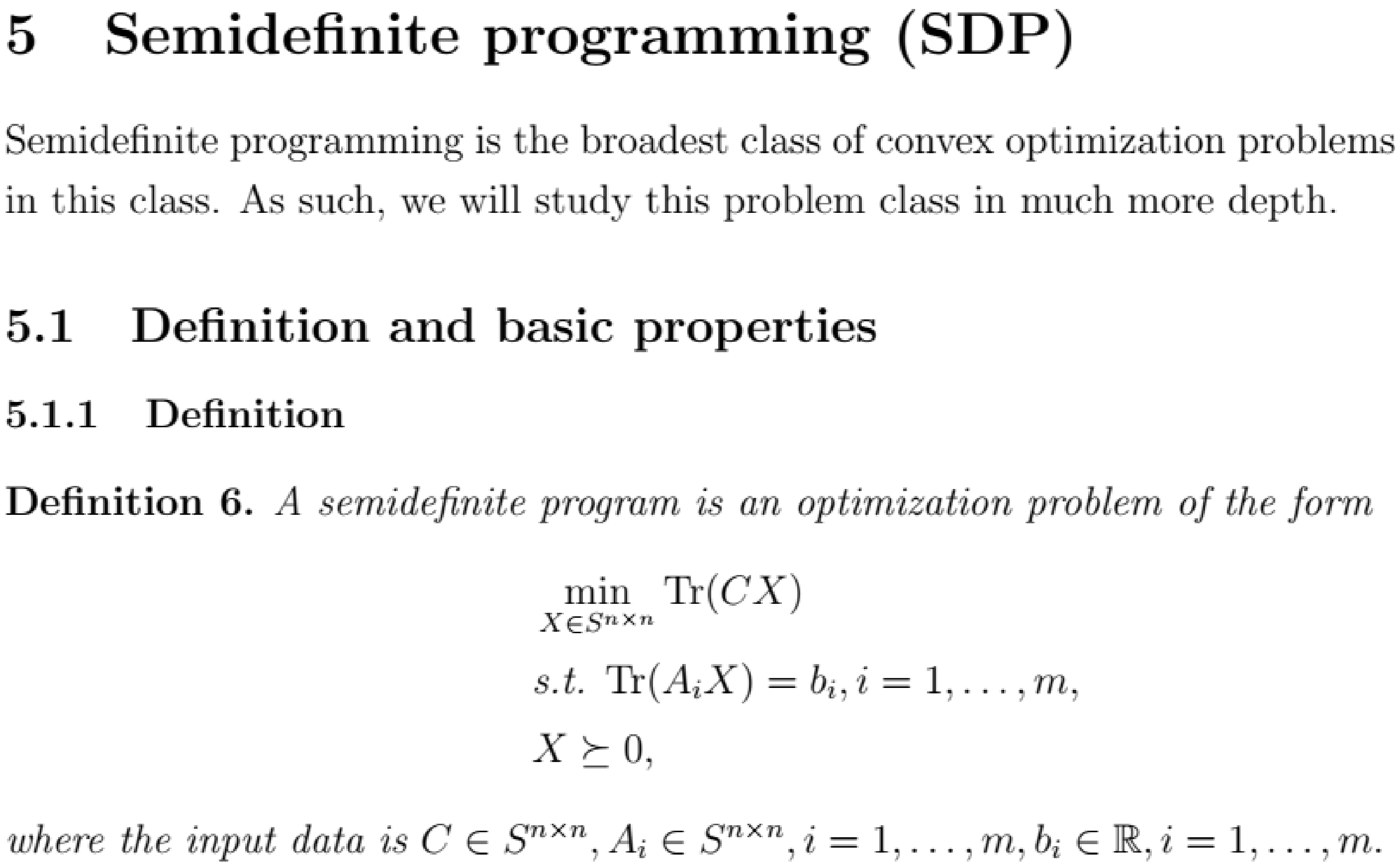
Definition 4. *A* second order cone program (SOCP) *is an optimization problem of the form:*

$$\min_x f^T x \tag{1}$$
$$||A_i x + b_i||_2 \leq c_i^T x + d_i, \ i = 1, \ldots, m,$$

where $A_i \in \mathbb{R}^{k_i \times n}$, $b_i \in \mathbb{R}^{k_i}$, $c_i \in \mathbb{R}^n$ *and* $d_i \in \mathbb{R}$.

The terminology of "SOCP" comes from its connection to *the second order cone* (also called the Lorentz cone or the ice-cream cone).

Julia Programming Language for Optimization



5 Semidefinite programming (SDP)

Semidefinite programming is the broadest class of convex optimization problems we consider in this class. As such, we will study this problem class in much more depth.

5.1 Definition and basic properties

5.1.1 Definition

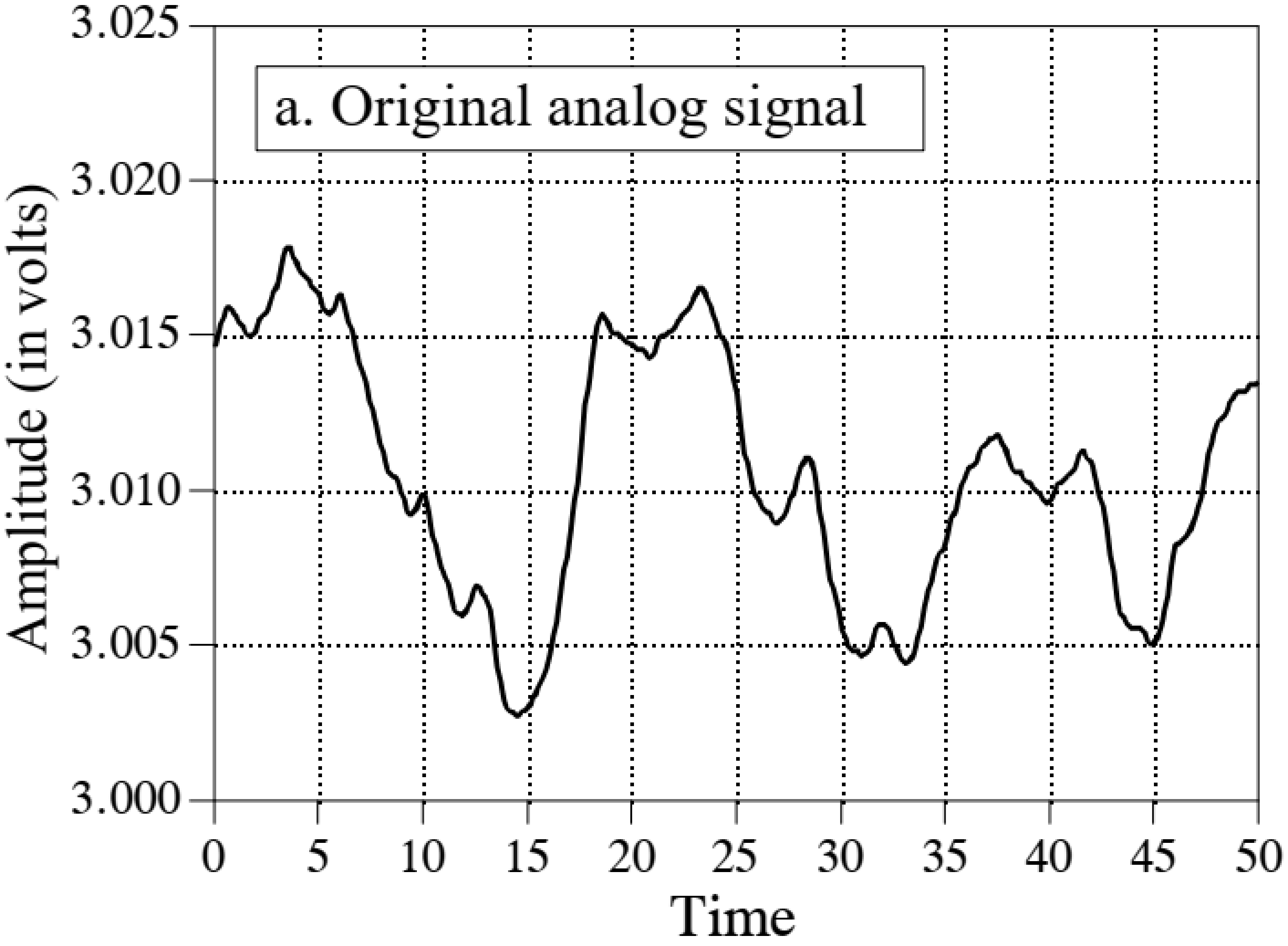
Definition 6. *A semidefinite program is an optimization problem of the form*

$$\begin{aligned}
&\min_{X \in S^{n \times n}} \operatorname{Tr}(CX) \\
&s.t. \ \operatorname{Tr}(A_i X) = b_i, i = 1, \ldots, m, \\
&X \succeq 0,
\end{aligned}$$

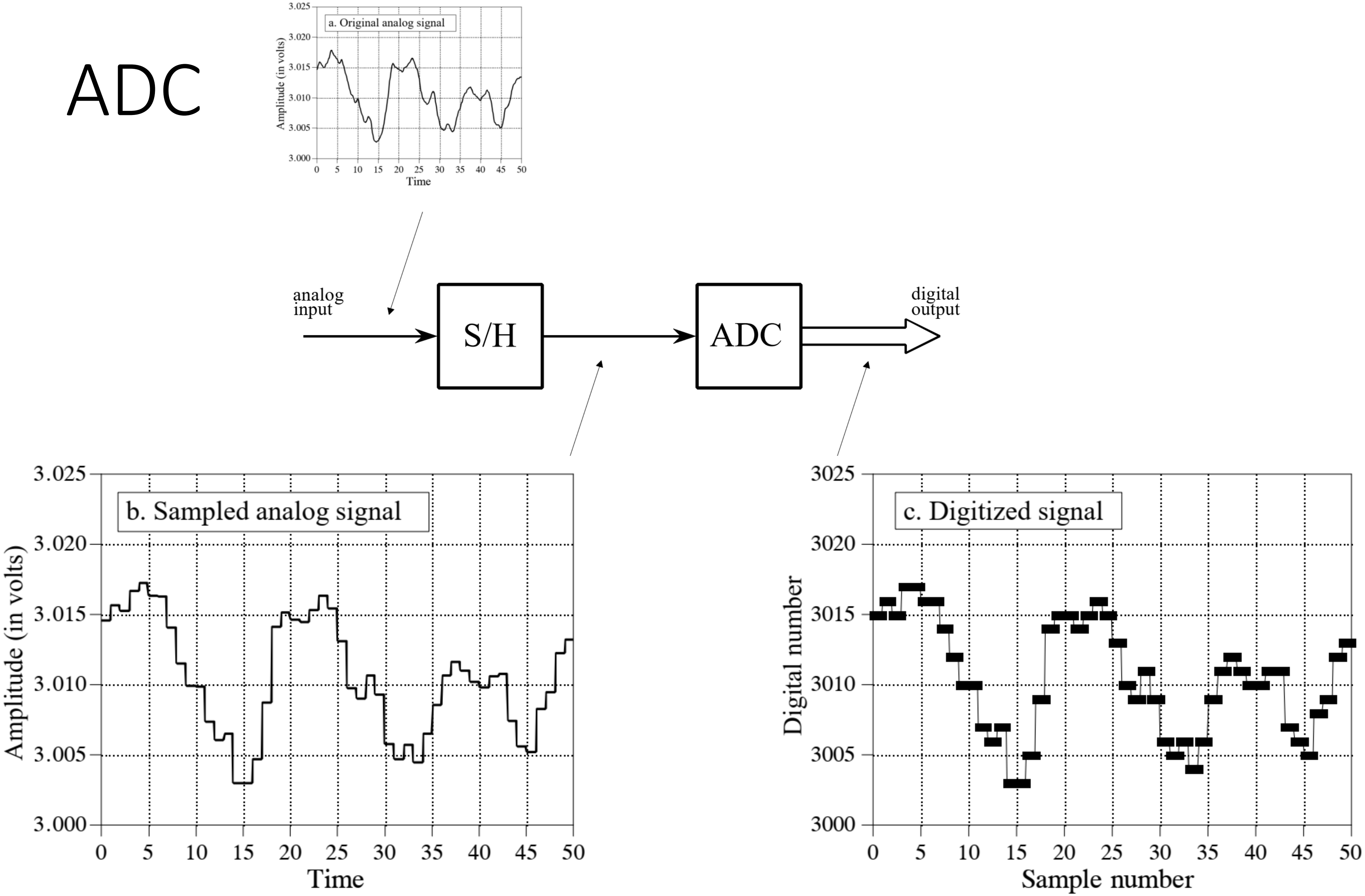
where the input data is $C \in S^{n \times n}, A_i \in S^{n \times n}, i = 1, \ldots, m, b_i \in \mathbb{R}, i = 1, \ldots, m.$

ADC

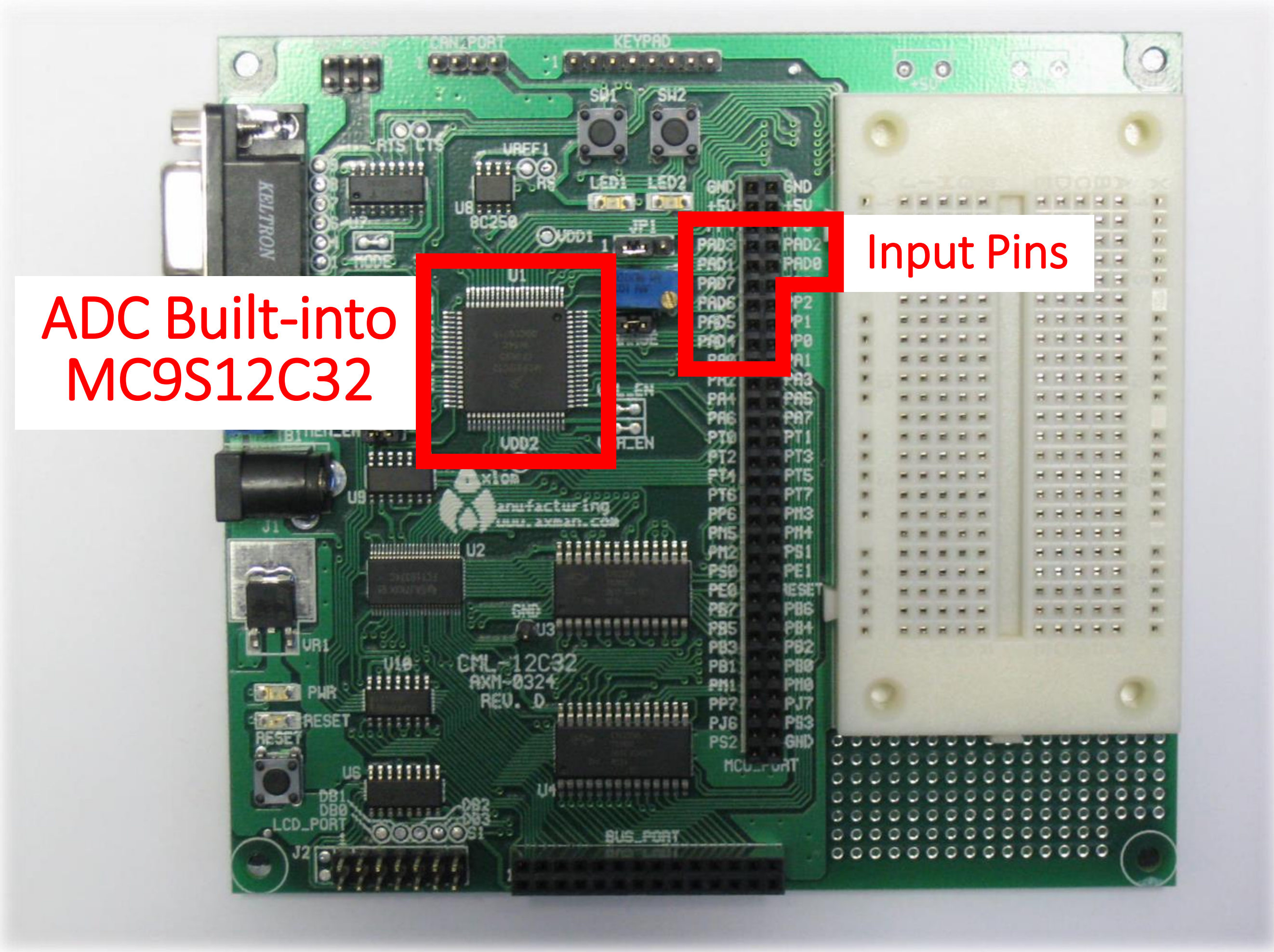
ADC


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

ADC


a. Original analog signal
Amplitude (in volts)
3.025
3.020
3.015
3.010
3.005
3.000
0 5 10 15 20 25 30 35 40 45 50
Time
analog input
S/H
ADC
digital output
b. Sampled analog signal
Amplitude (in volts)
3.025
3.020
3.015
3.010
3.005
3.000
0 5 10 15 20 25 30 35 40 45 50
Time
c. Digitized signal
Digital number
3025
3020
3015
3010
3005
3000
0 5 10 15 20 25 30 35 40 45 50
Sample number

ADC



ADC Built-into MC9S12C32

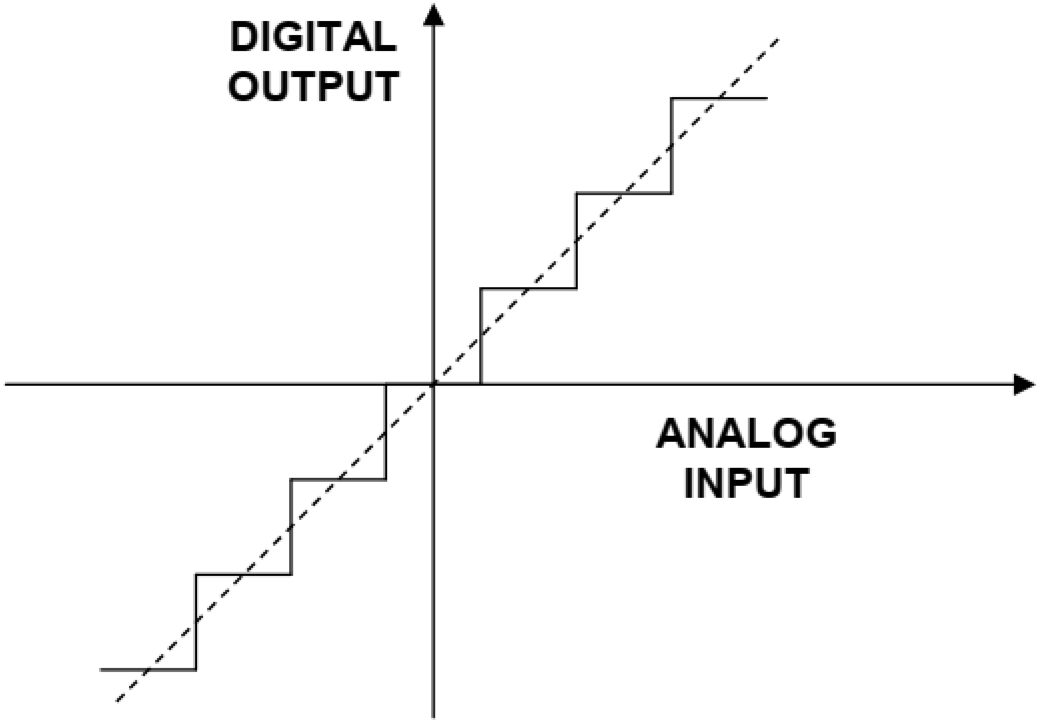
Input Pins

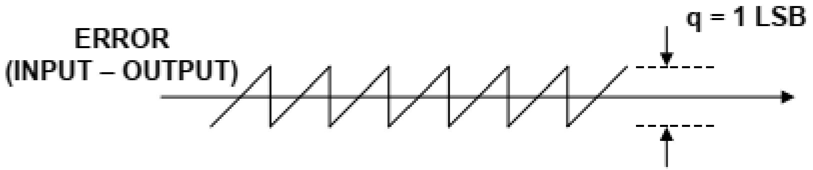
ADC

ADC Sampling Rate (500MHz)

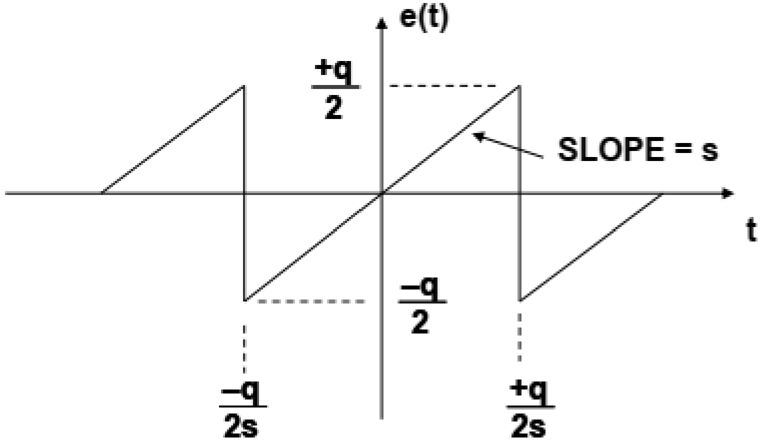
ADC Quantization Bits (12 bit)

ADC Quantization Noise





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



ADC Quantization Noise

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

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

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

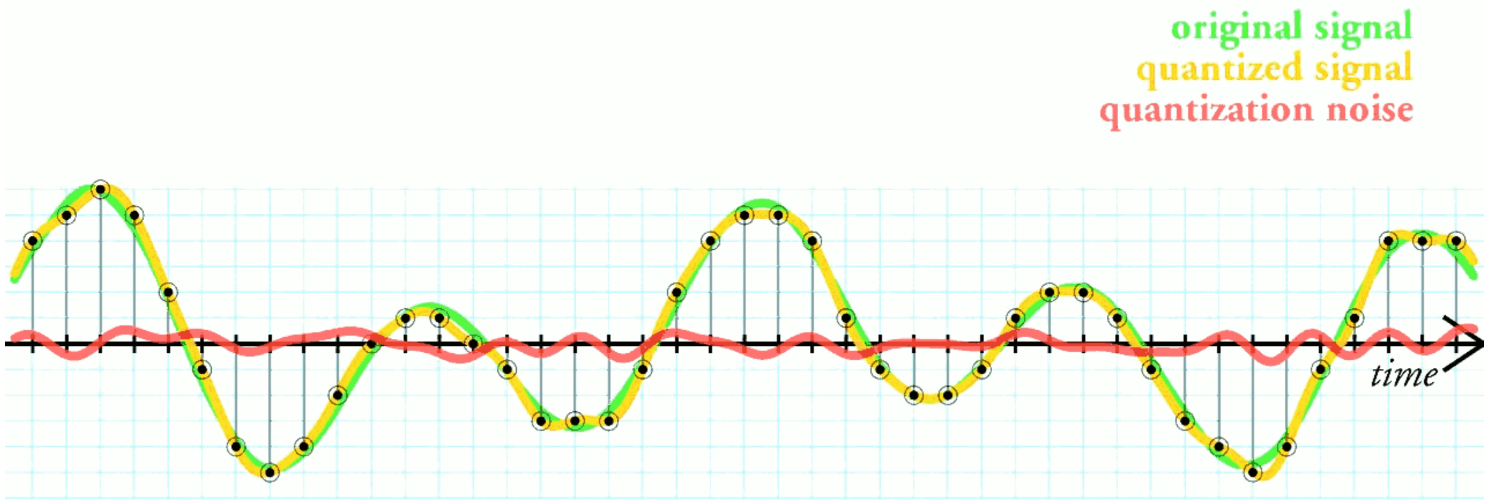
ADC Quantization Noise

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

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

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

ADC Quantization SNR (N Bit)



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

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

ADC Quantization SNR (N Bit)

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

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

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

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

ADC Quantization SNR (N Bit)

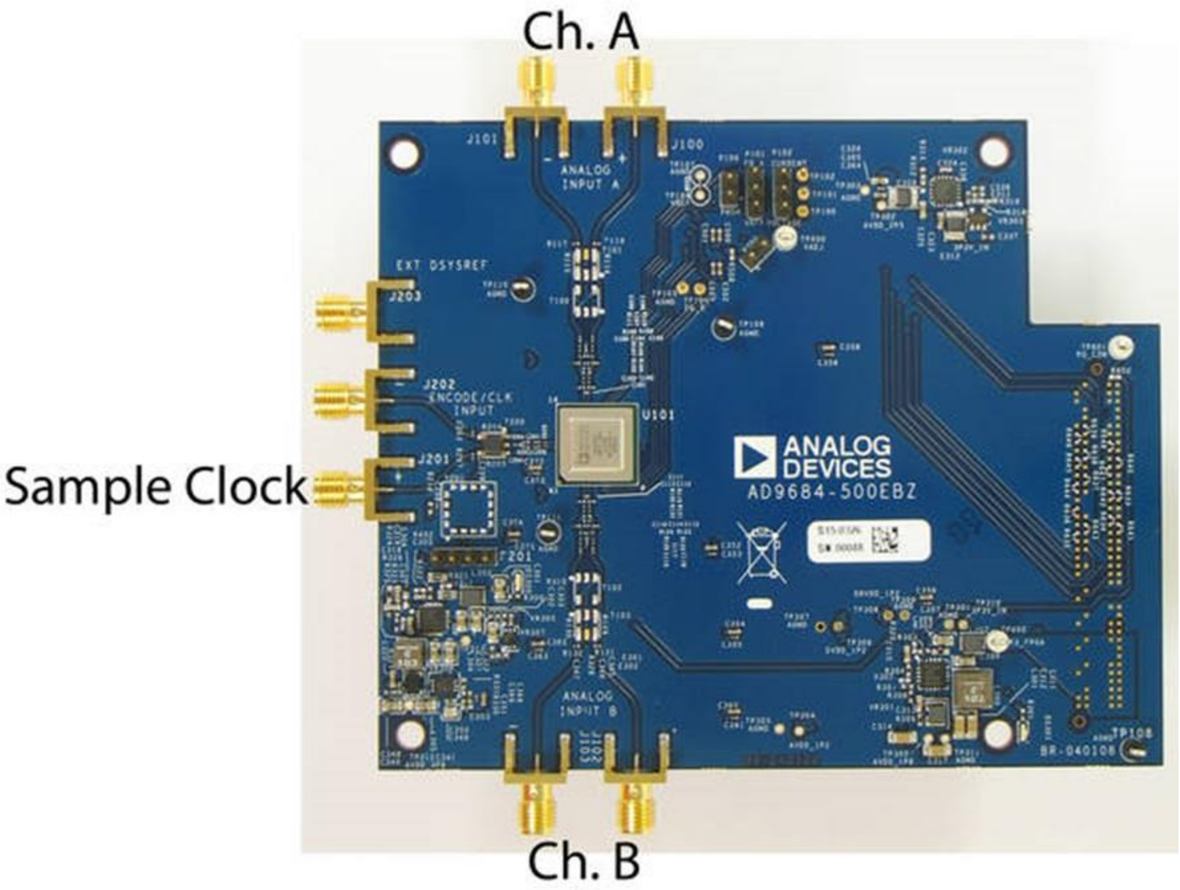
$$\mathrm{SNR} = 6.02N + 1.76\mathrm{dB}$$

- N = 2 SNR = 13.8 dB
- N = 3 SNR = 19.8 dB
- N = 6 SNR = 37.9 dB
- N = 8 SNR = 49.9 dB
- N = 10 SNR = 61.96 dB

- N = 12 SNR = 74 dB
- N = 14 SNR = 86 dB
- N = 16 SNR = 98 dB

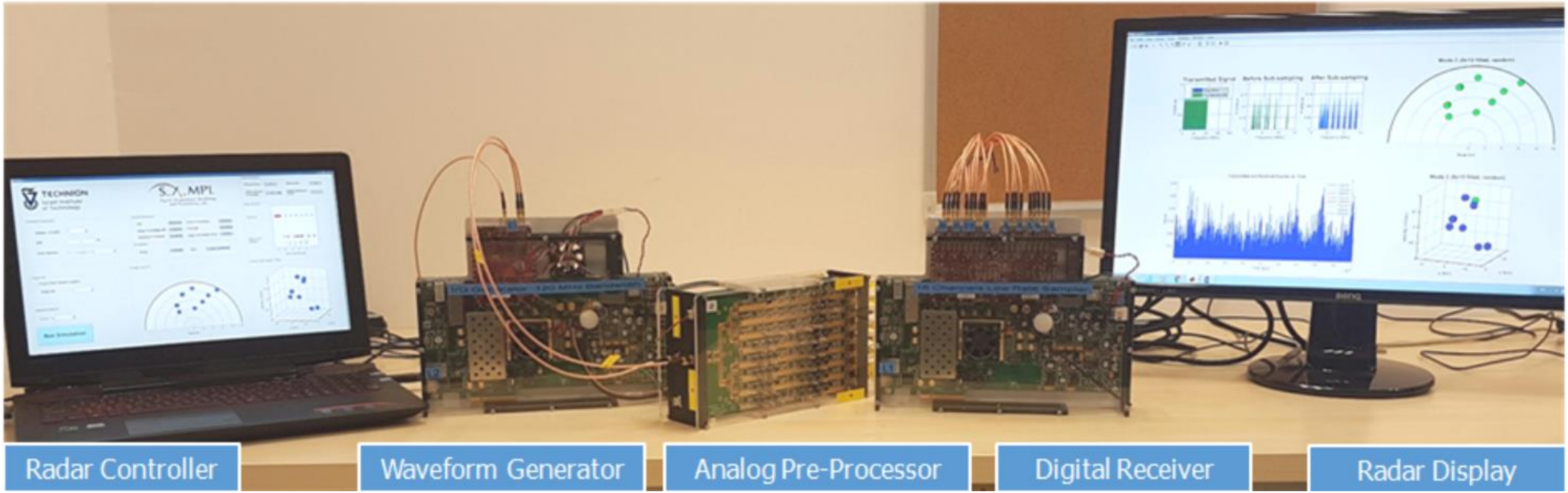
ADC

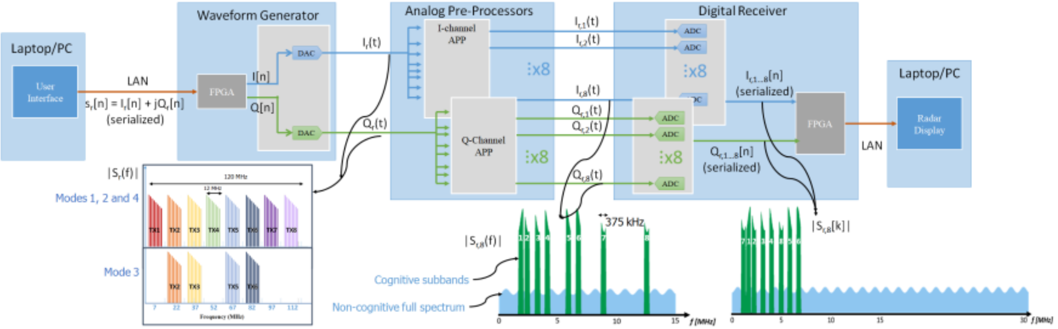
AD9684



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

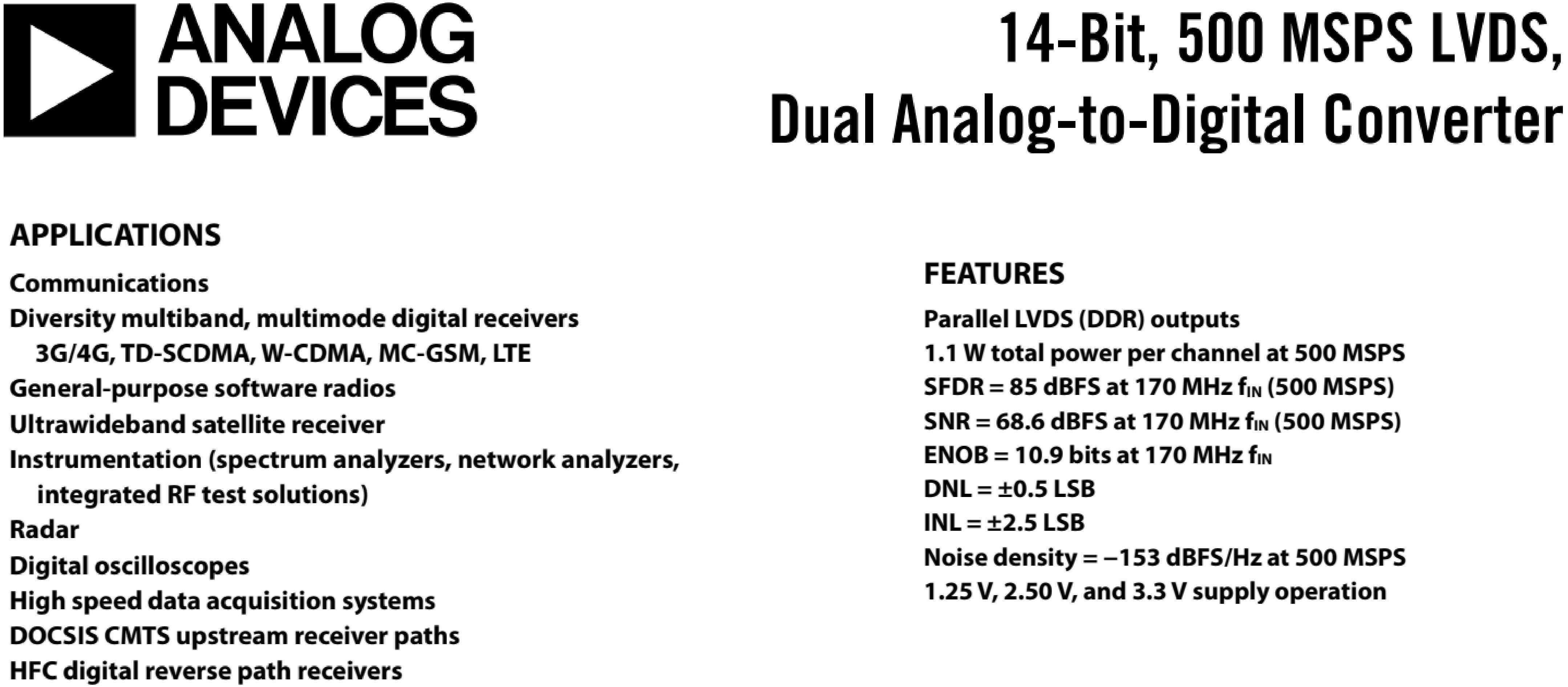
ADC in SUMMeR (Sub-Nyquist colocated MiMo Radar)





ADC

AD9684



ANALOG DEVICES

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

APPLICATIONS

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

FEATURES

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

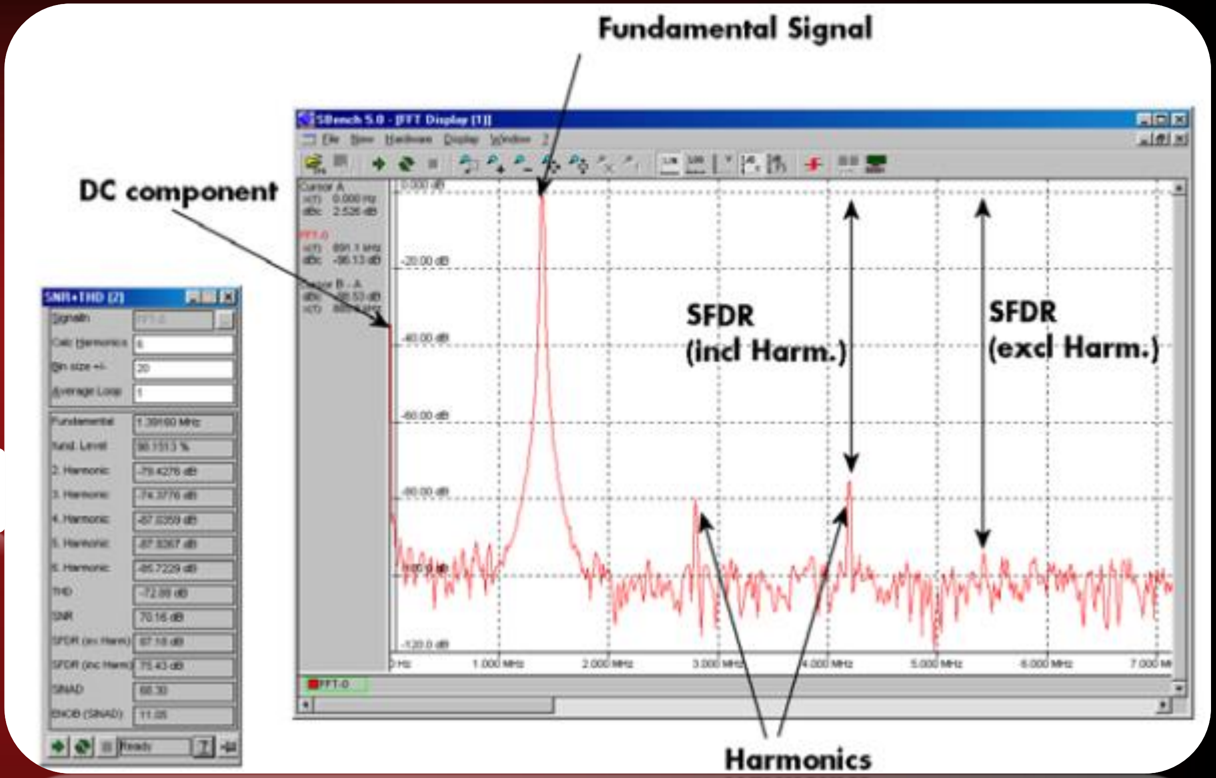
ADC

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

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

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

ENOB = (SINAD – 1.76) / 6.02

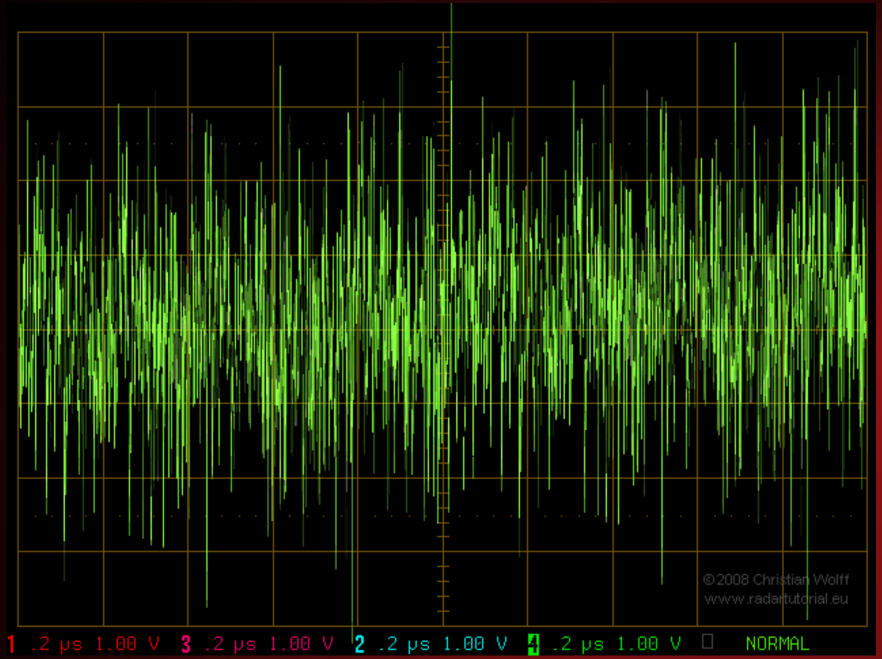


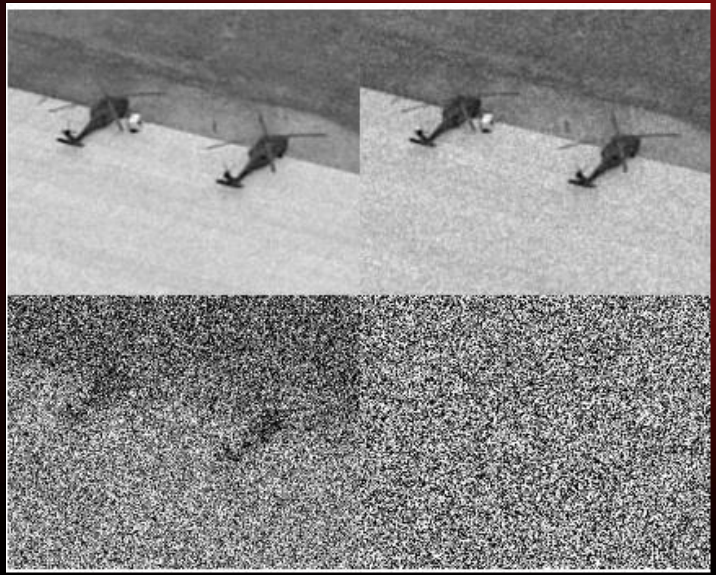
Signal	FFT-0
Calc Harmonics	6
Bin size +/-	20
Average Loop	1
Fundamental	1.39160 MHz
fund. Level	98.1513 %
2. Harmonic	-79.4276 dB
3. Harmonic	-74.3776 dB
4. Harmonic	-87.0359 dB
5. Harmonic	-87.8367 dB
6. Harmonic	-85.7229 dB
THD	-72.99 dB
SNR	70.16 dB
SFDR (ex Harm)	87.18 dB
SFDR (inc Harm)	75.43 dB
SINAD	68.30
ENOB (SINAD)	11.05

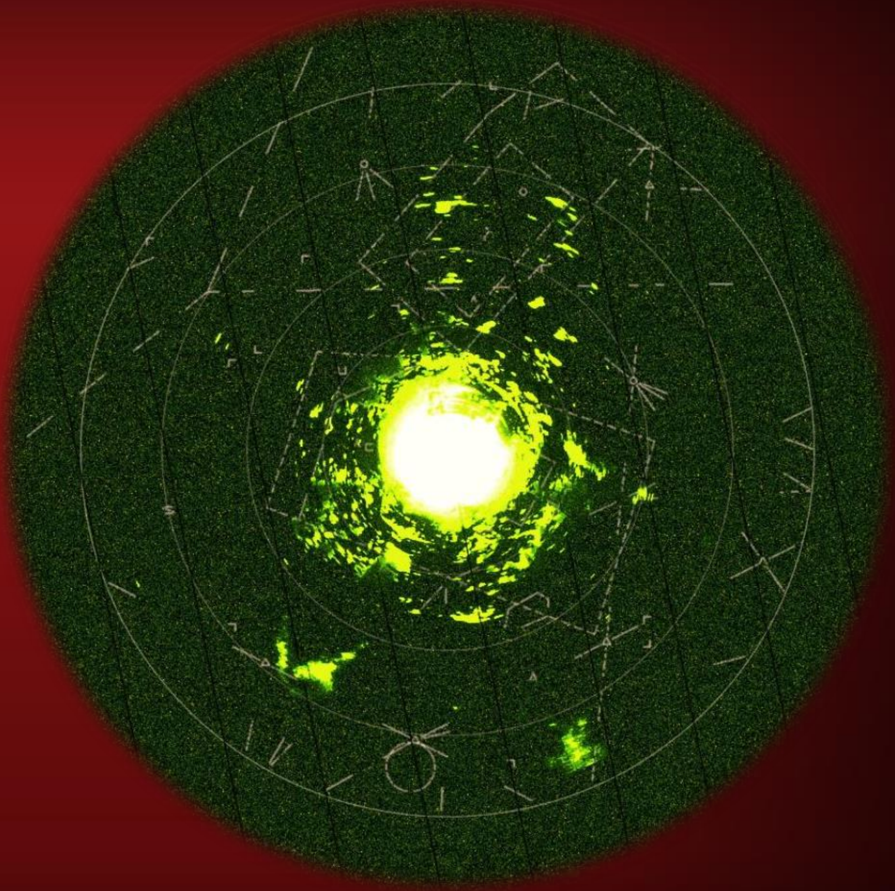
Ready

Noise

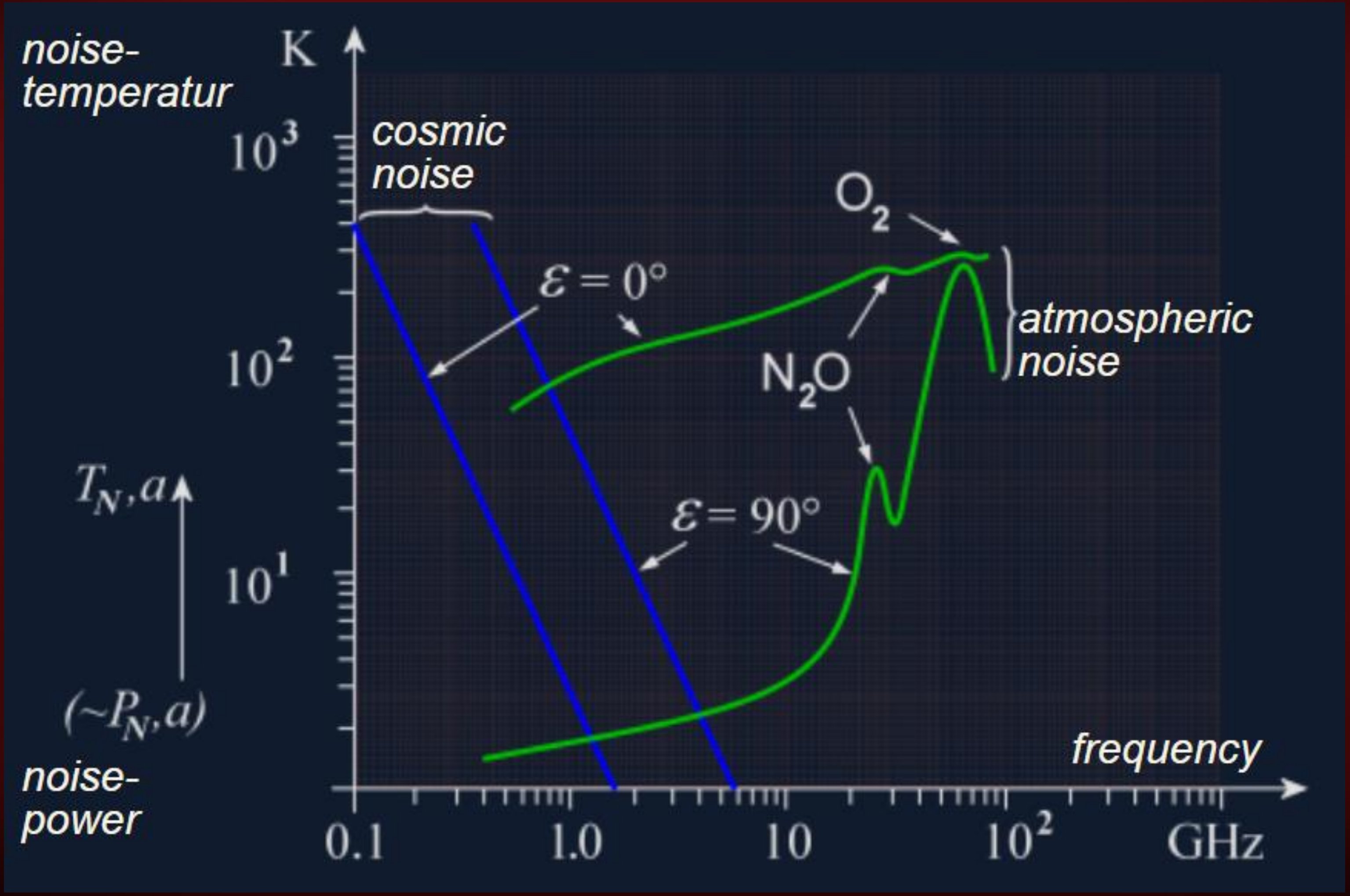
Noise


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



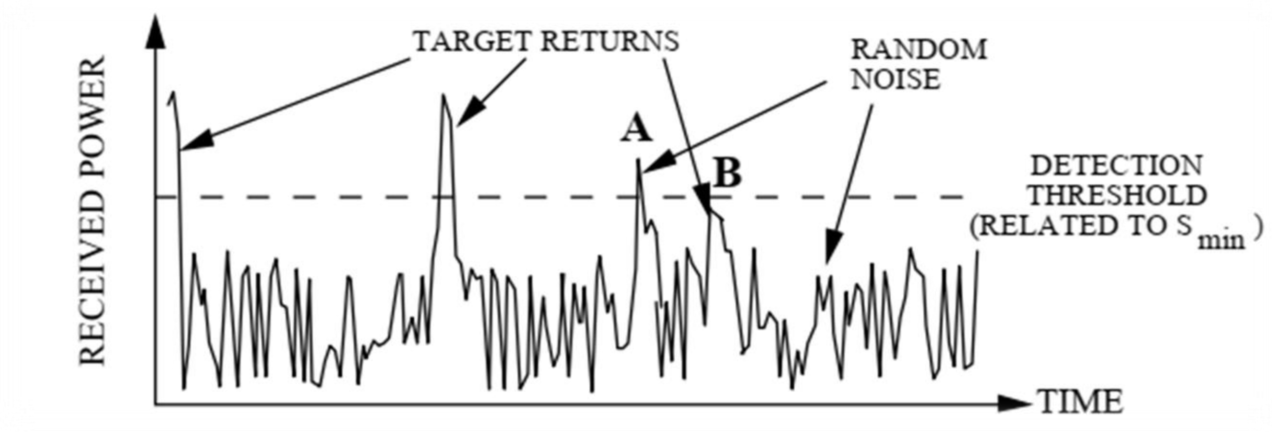


Noise



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

Thermal Noise



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

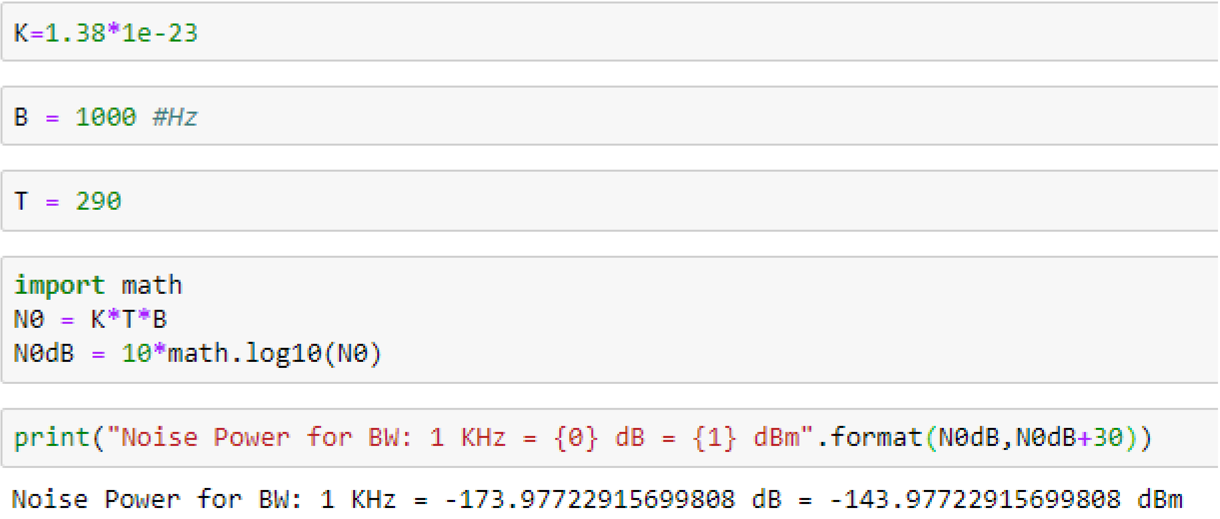
$$N_o = k T_o B_n$$

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

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

Thermal Noise

$$N_o = kT_oB_n$$



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

```
B = 1000 #Hz
```

```
T = 290
```

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

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

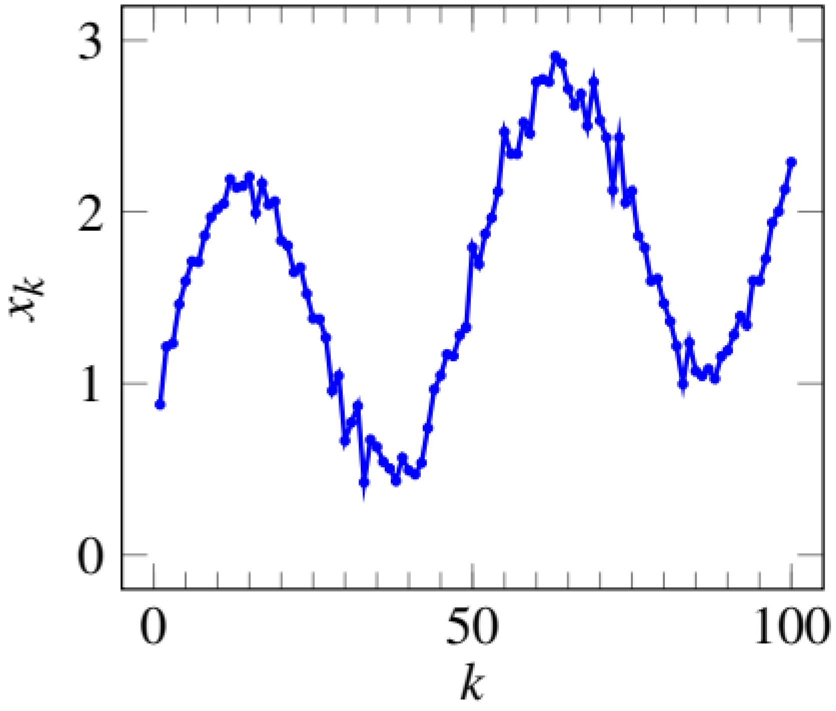
```
Noise Power for BW: 1 KHz = -173.97722915699808 dB = -143.97722915699808 dBm
```

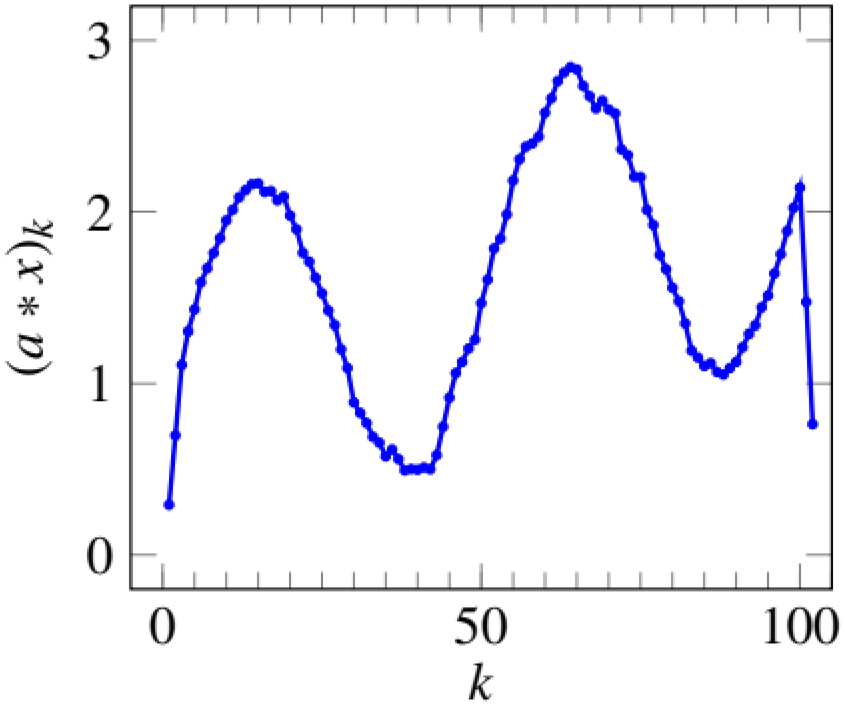
System Noise

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

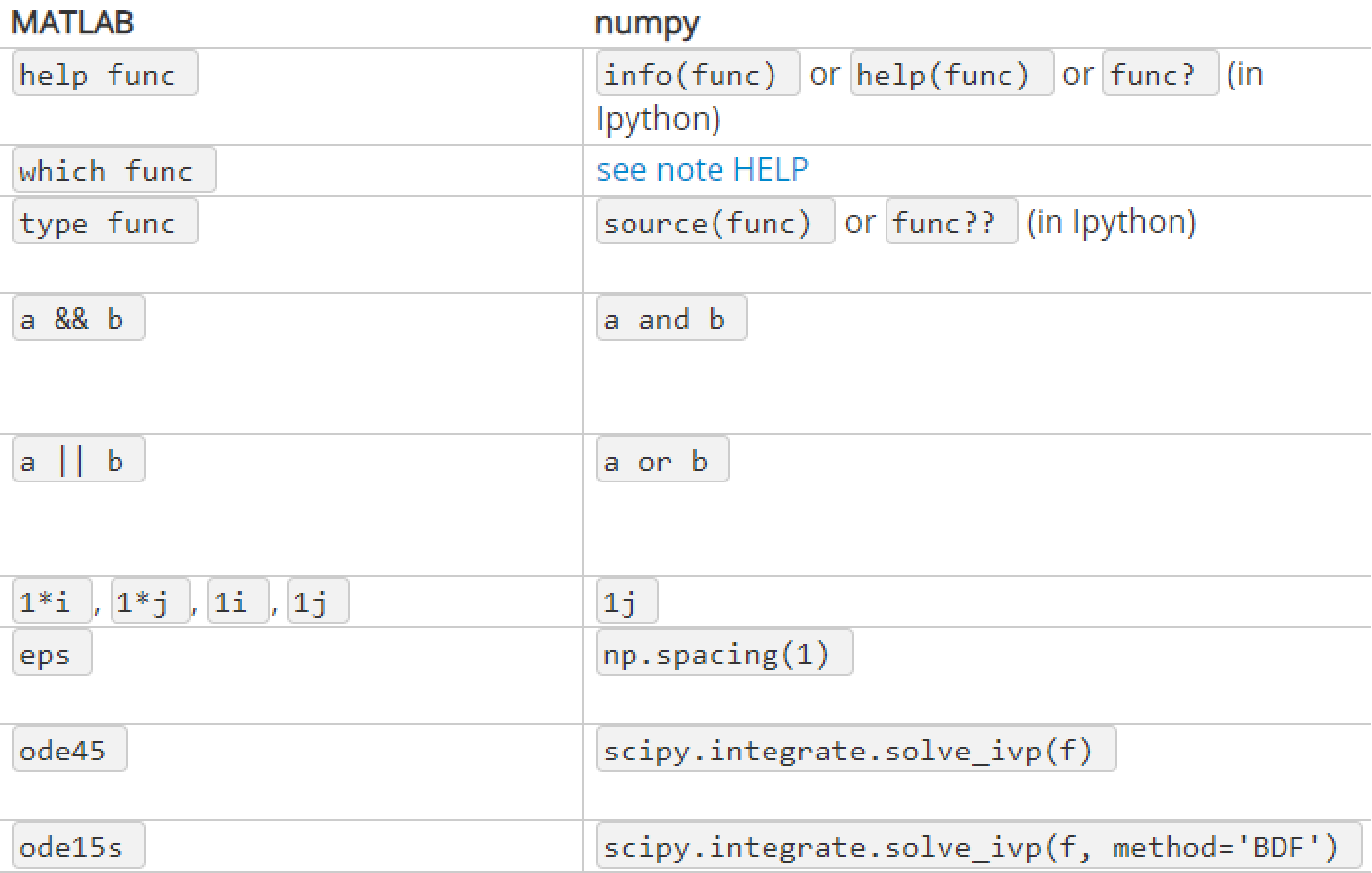
Moving Average Filter

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

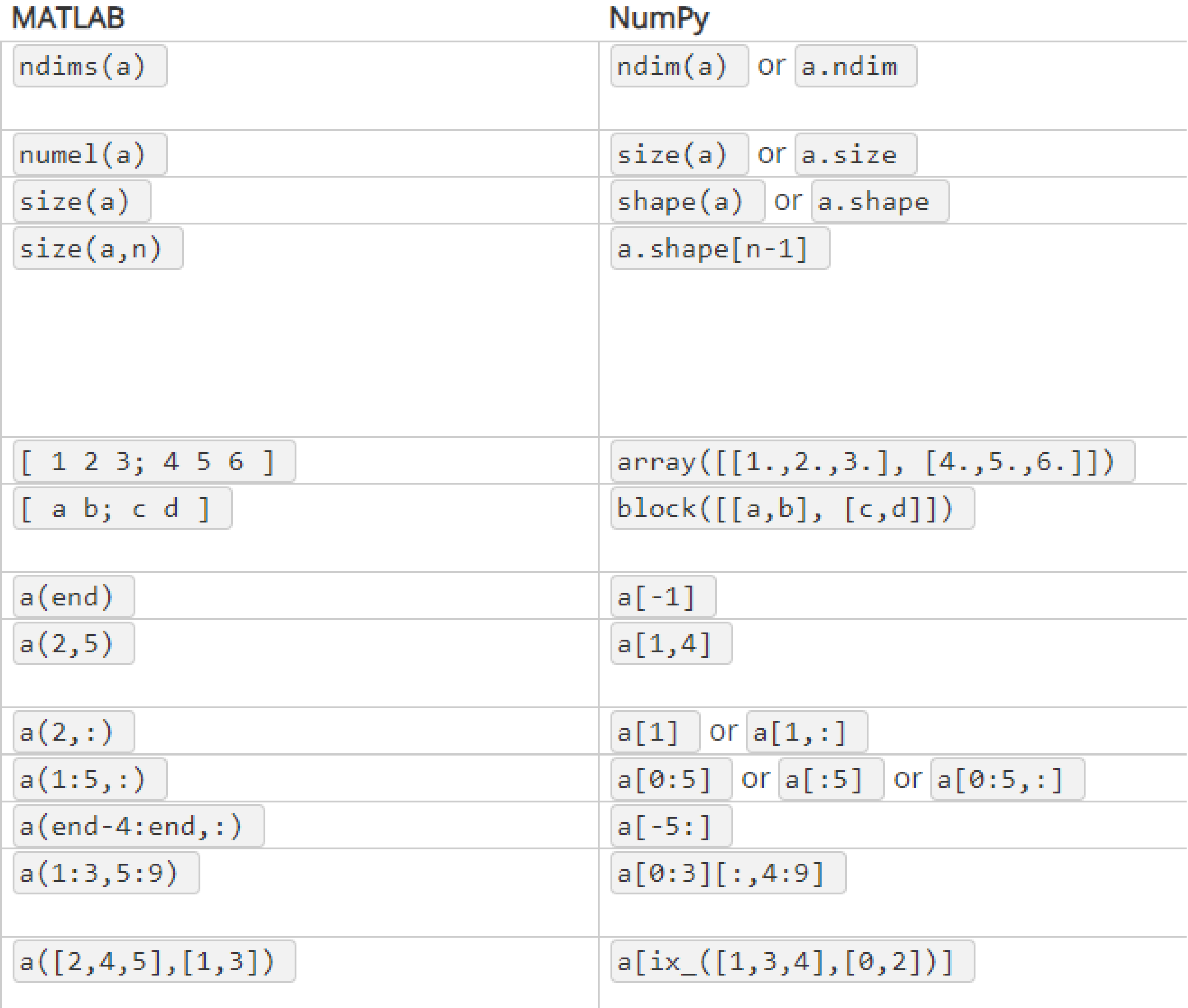




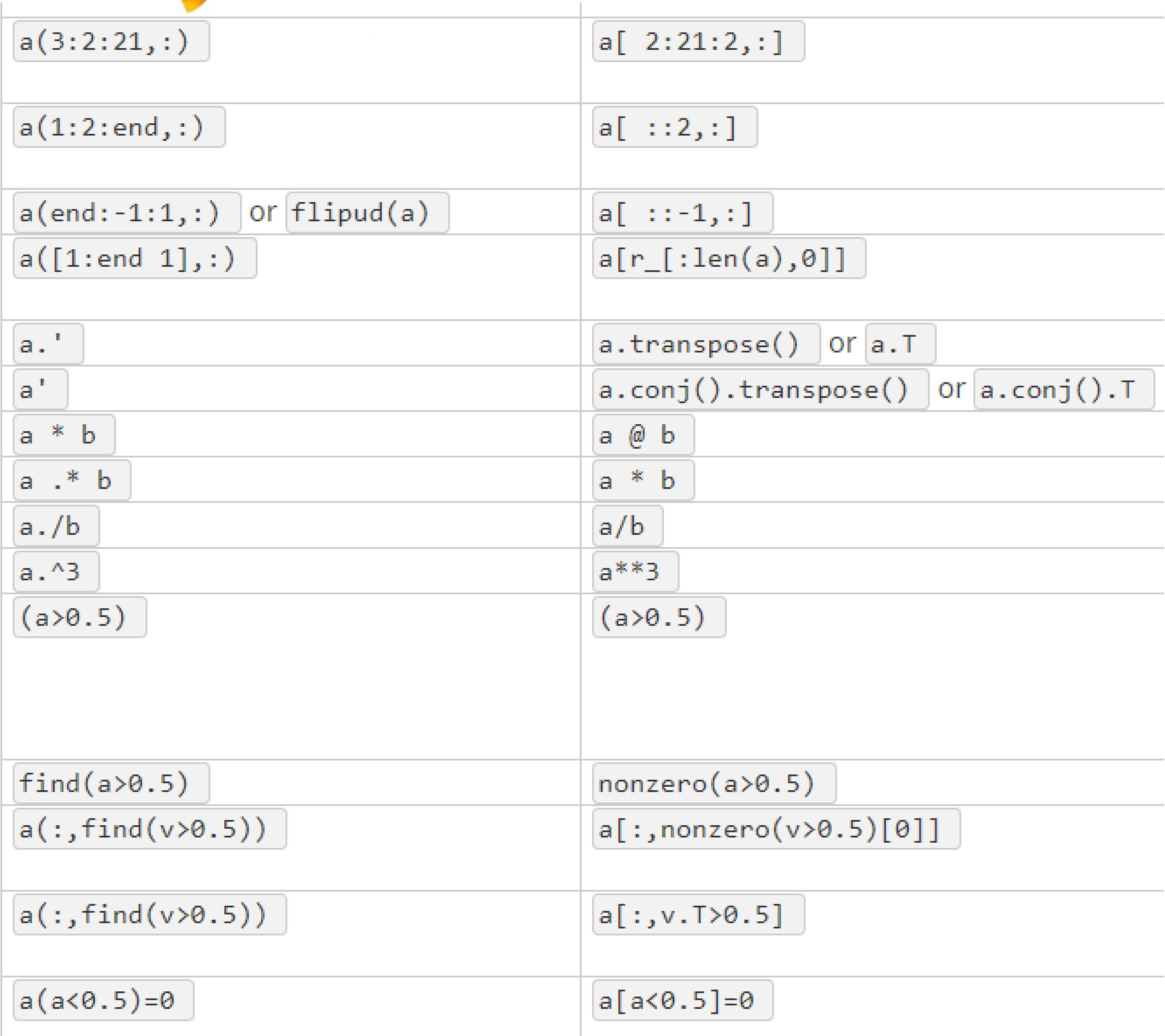
Python Numpy & MATLAB



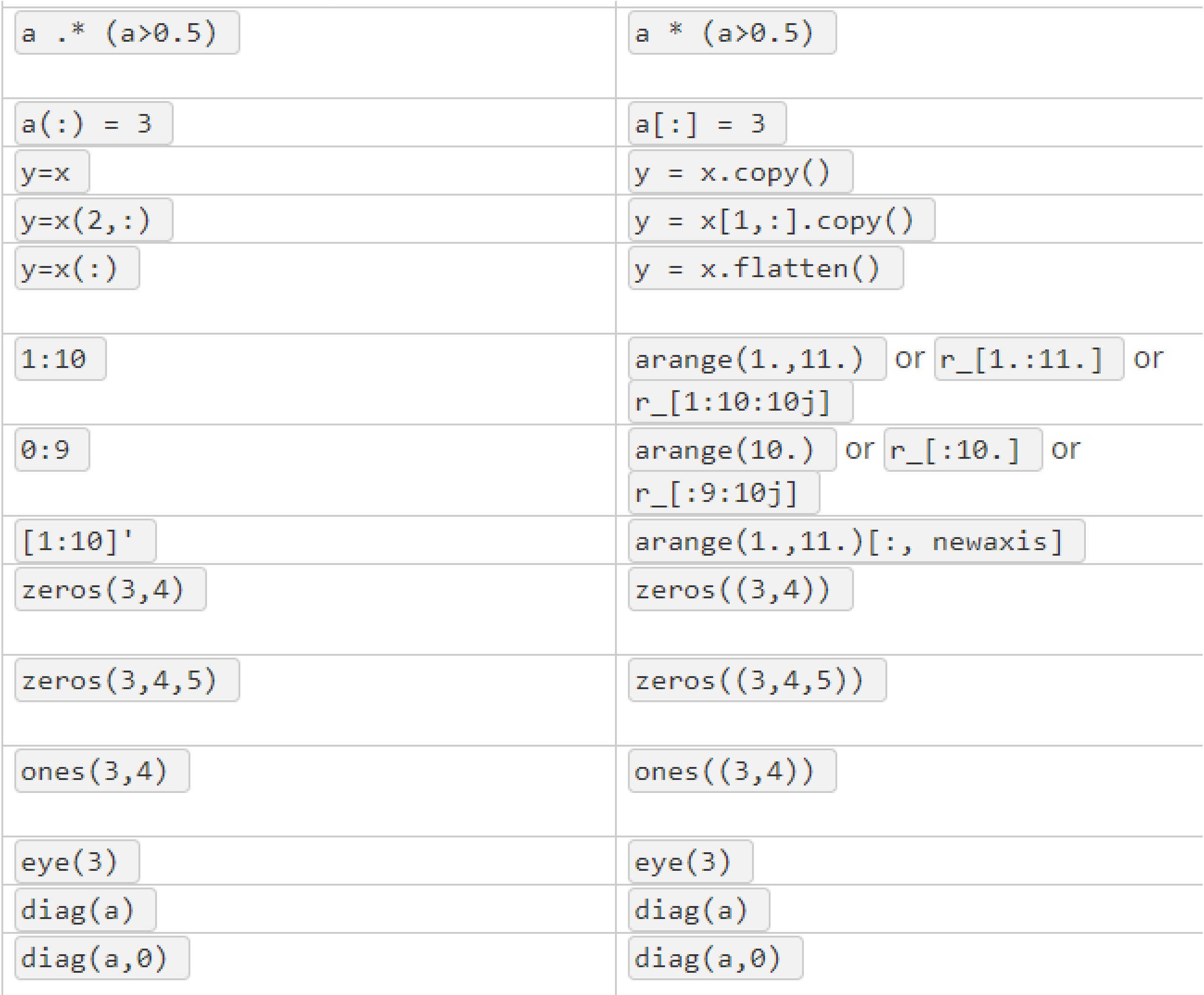
MATLAB	numpy
`help func`	`info(func)` or `help(func)` or `func?` (in Ipython)
`which func`	see note HELP
`type func`	`source(func)` or `func??` (in Ipython)
`a && b`	`a and b`
`a || b`	`a or b`
`1*i`, `1*j`, `1i`, `1j`	`1j`
`eps`	`np.spacing(1)`
`ode45`	`scipy.integrate.solve_ivp(f)`
`ode15s`	`scipy.integrate.solve_ivp(f,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



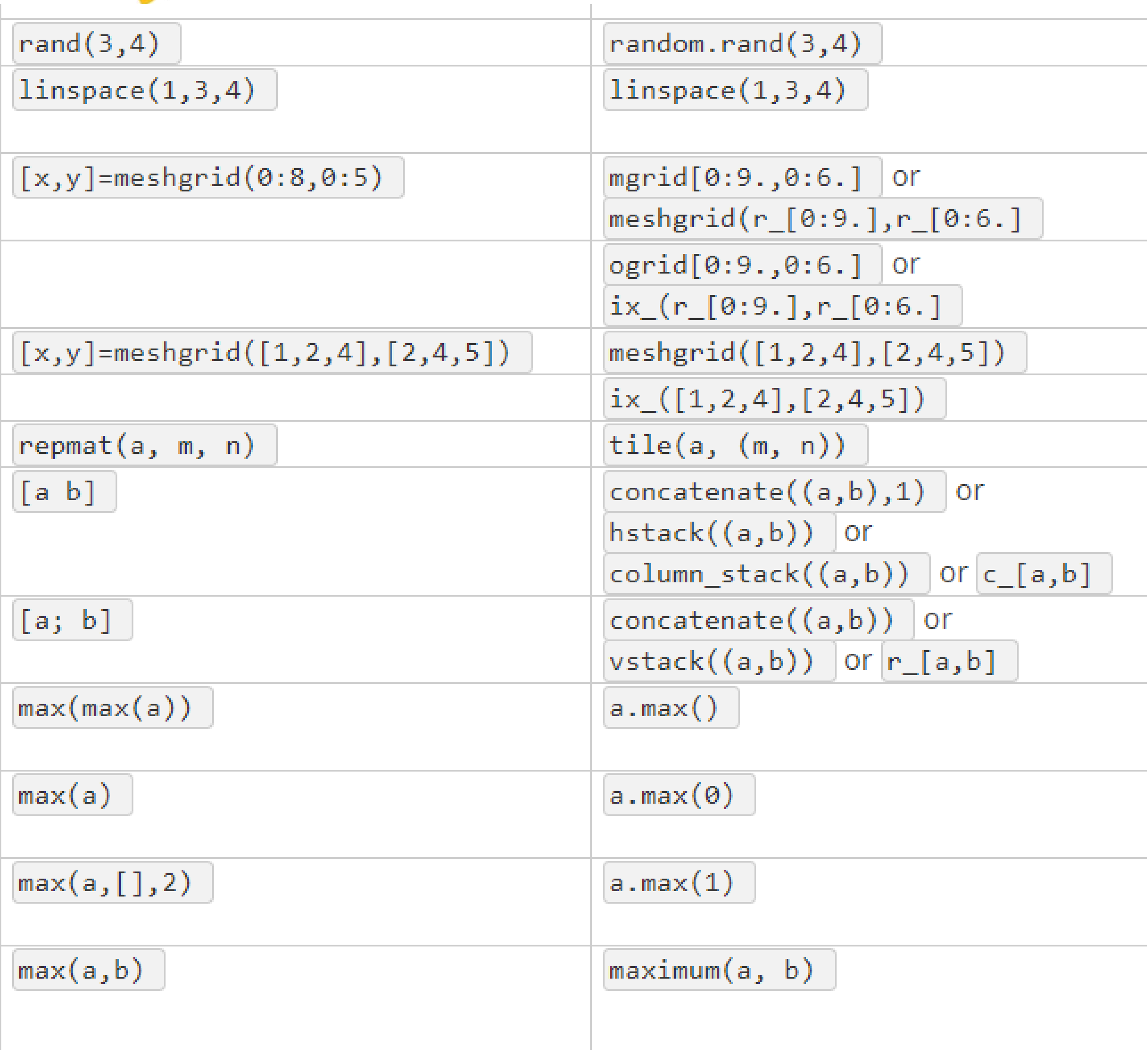
MATLAB	NumPy
`ndims(a)`	`ndim(a)` or `a.ndim`
`numel(a)`	`size(a)` or `a.size`
`size(a)`	`shape(a)` or `a.shape`
`size(a,n)`	`a.shape[n-1]`
`[ 1 2 3; 4 5 6 ]`	`array([[1.,2.,3.], [4.,5.,6.]])`
`[ a b; c d ]`	`block([[a,b], [c,d]])`
`a(end)`	`a[-1]`
`a(2,5)`	`a[1,4]`
`a(2,:)`	`a[1]` or `a[1,:]`
`a(1:5,:)`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



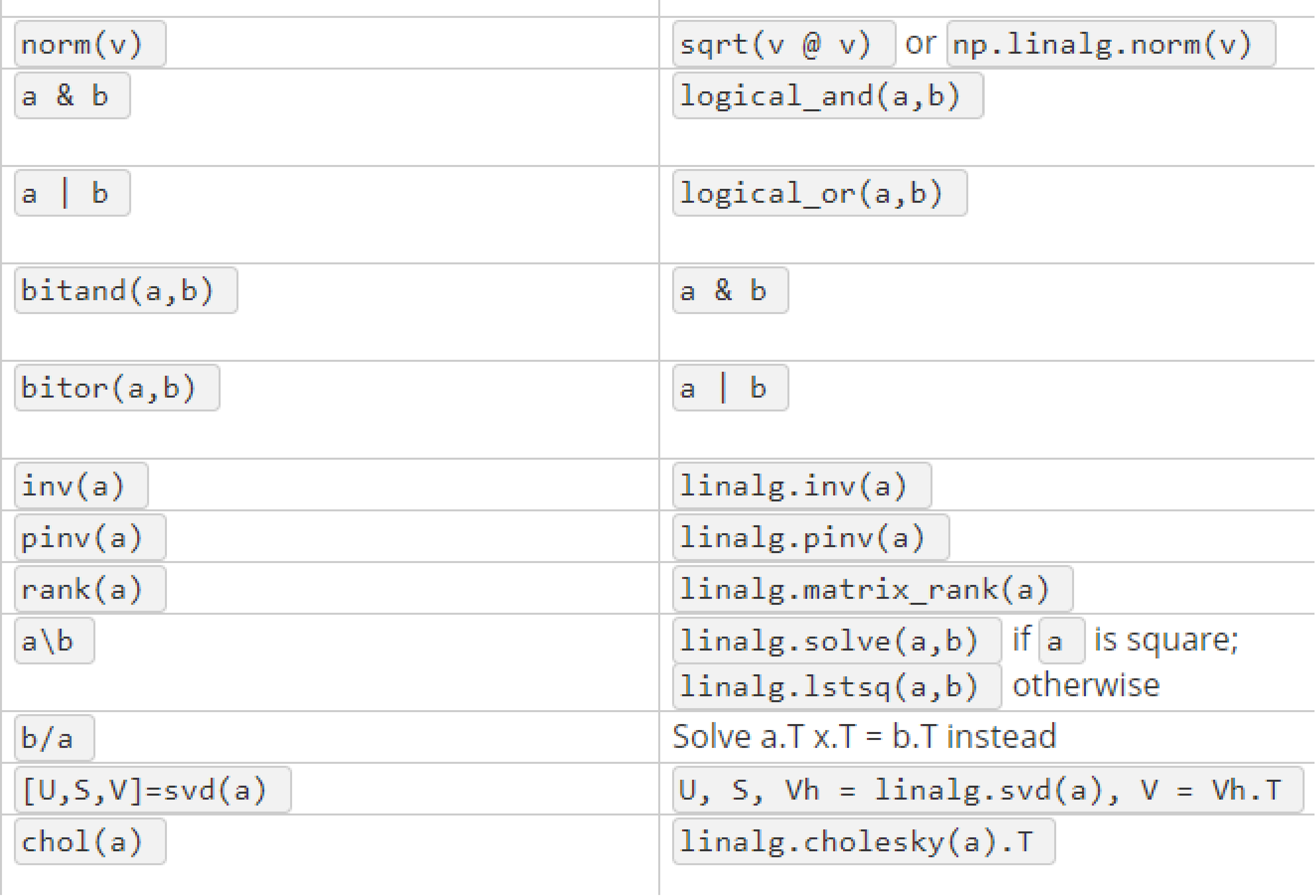
MATLAB	python
`a(3:2:21,:)`	`a[ 2:21:2,:]`
`a(1:2:end,:)`	`a[ ::2,:]`
`a(end:-1:1,:)` or `flipud(a)`	`a[ ::-1,:]`
`a([1:end 1],:)`	`a[r_[:len(a),0]]`
`a.'`	`a.transpose()` or `a.T`
`a'`	`a.conj().transpose()` or `a.conj().T`
`a * b`	`a @ b`
`a .* b`	`a * b`
`a./b`	`a/b`
`a.^3`	`a**3`
`(a>0.5)`	`(a>0.5)`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



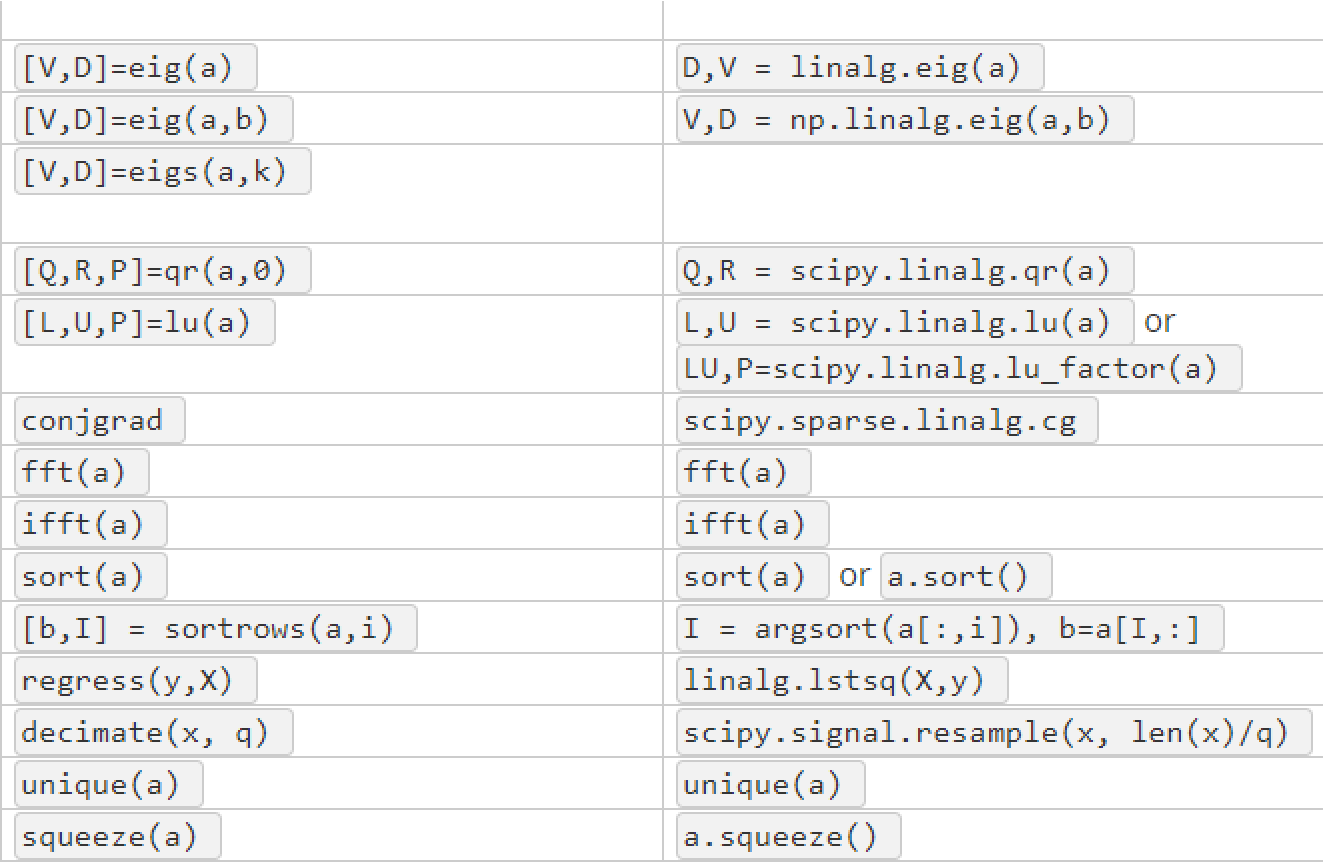
MATLAB	python™
`a .* (a>0.5)`	`a * (a>0.5)`
`a(:) = 3`	`a[:] = 3`
`y=x`	`y = x.copy()`
`y=x(2,:)`	`y = x[1,:].copy()`
`y=x(:)`	`y = x.flatten()`
`1:10`	`arange(1.,11.)` or `r_[1.:11.]` or `r_[1:10:10j]`
`0:9`	`arange(10.)` or `r_[:10.]` or `r_[:9:10j]`
`[1:10]'`	`arange(1.,11.)[:, newaxis]`
`zeros(3,4)`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



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



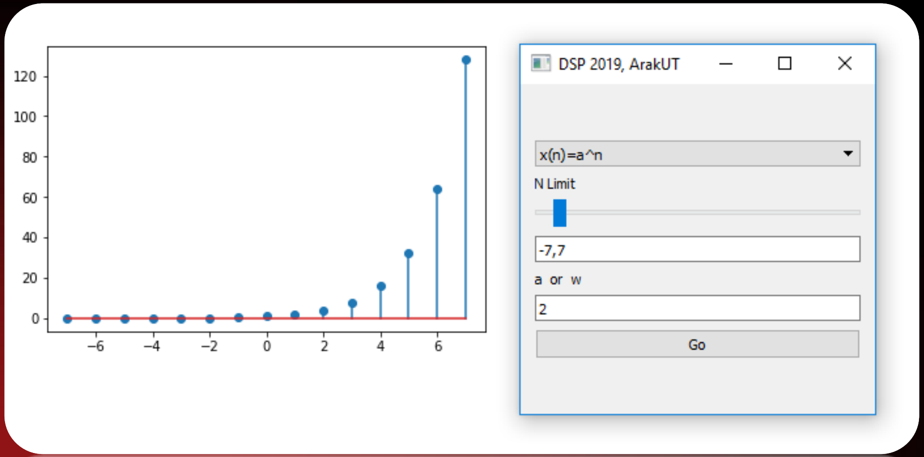
MATLAB	python™
`norm(v)`	`sqrt(v @ v)` or `np.linalg.norm(v)`
`a & b`	`logical_and(a,b)`
`a | b`	`logical_or(a,b)`
`bitand(a,b)`	`a & b`
`bitor(a,b)`	`a | b`
`inv(a)`	`linalg.inv(a)`
`pinv(a)`	`linalg.pinv(a)`
`rank(a)`	`linalg.matrix_rank(a)`
`a\b`	`linalg.solve(a,b)` if `a` is square; `linalg.lstsq(a,b)`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



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

Extra Programming Skills

DSP+Python UI



DSP 2019, ArakUT

x(n)=a^n

N Limit

-7,7

a or w

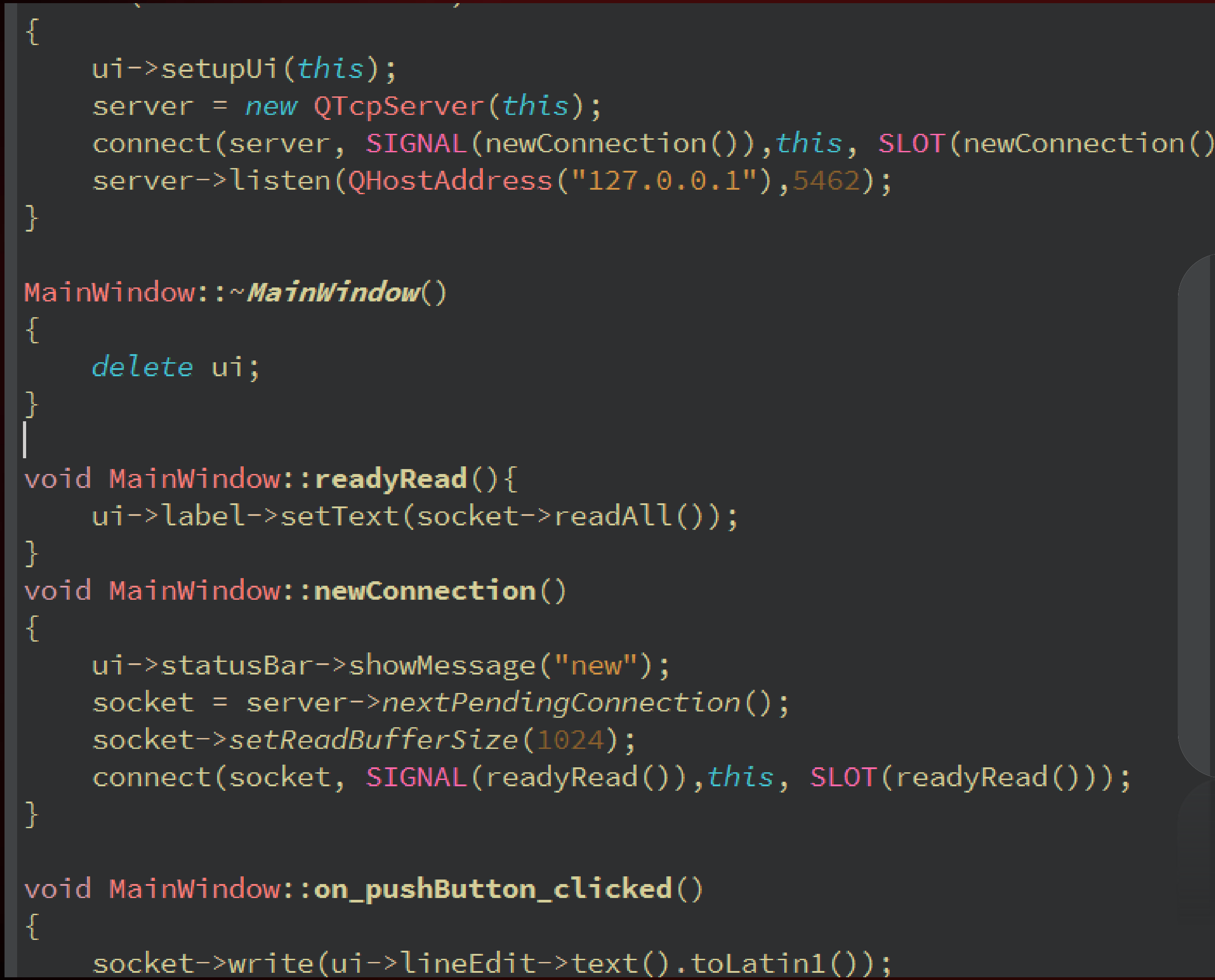
2

Go

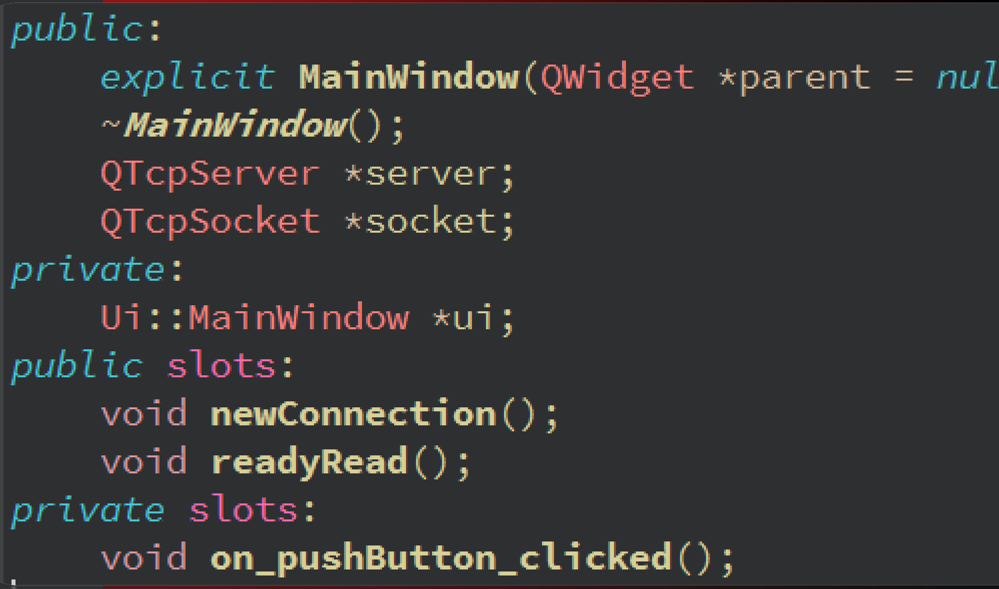
```
from PyQt5.QtWidgets import
QApplication,QWidget,QPushButton,QLineEdit,QSlider,QComboBox,QVBoxLayout,QHBoxLayout,QLabel
from PyQt5.QtCore import *
import matplotlib.pyplot as plt
import math
def sliderslot(value):
    dspLineEdit.setText(str(-(value+1))+","+str(value+1))
def pbslot():
    case =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

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

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

TCP Connection in Python and C++

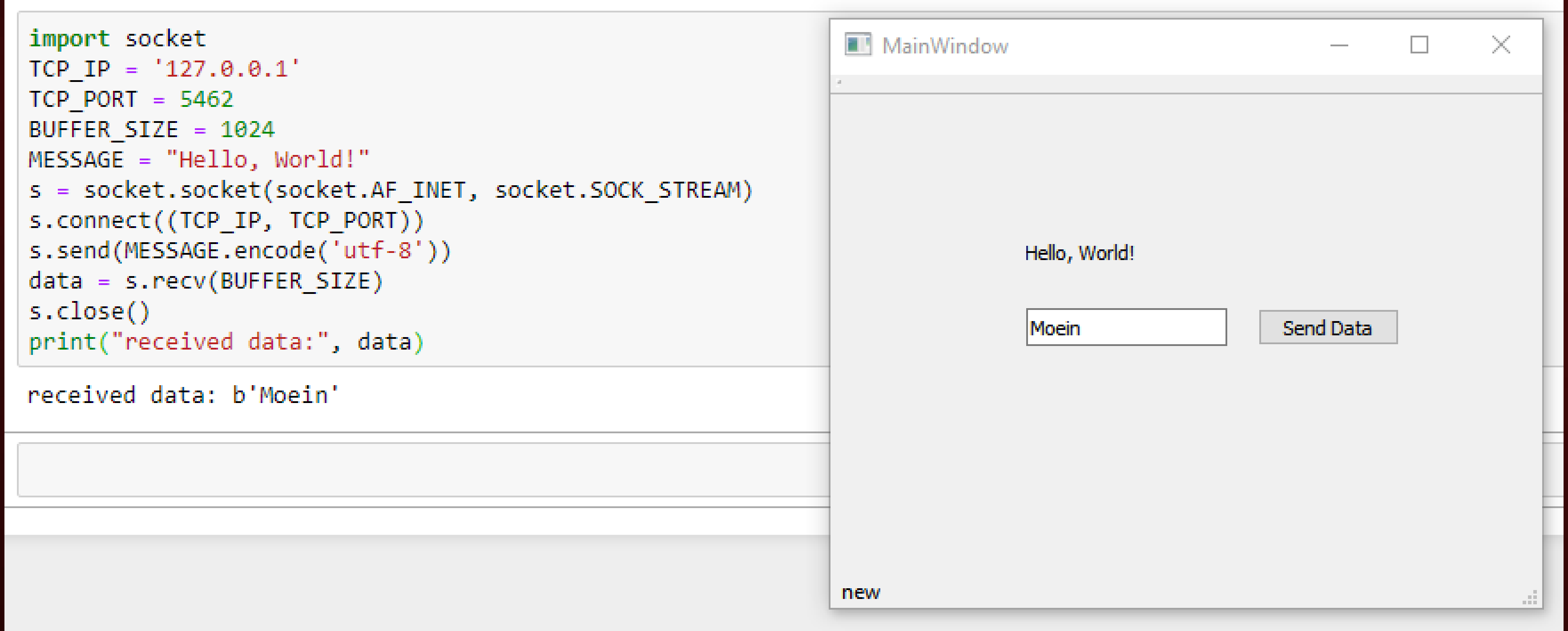


```
{
    ui->setupUi(this);
    server = new QTcpServer(this);
    connect(server, SIGNAL(newConnection()),this, SLOT(newConnection()));
    server->listen(QHostAddress("127.0.0.1"),5462);
}

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

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

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



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

TCP Connection in Python and C++



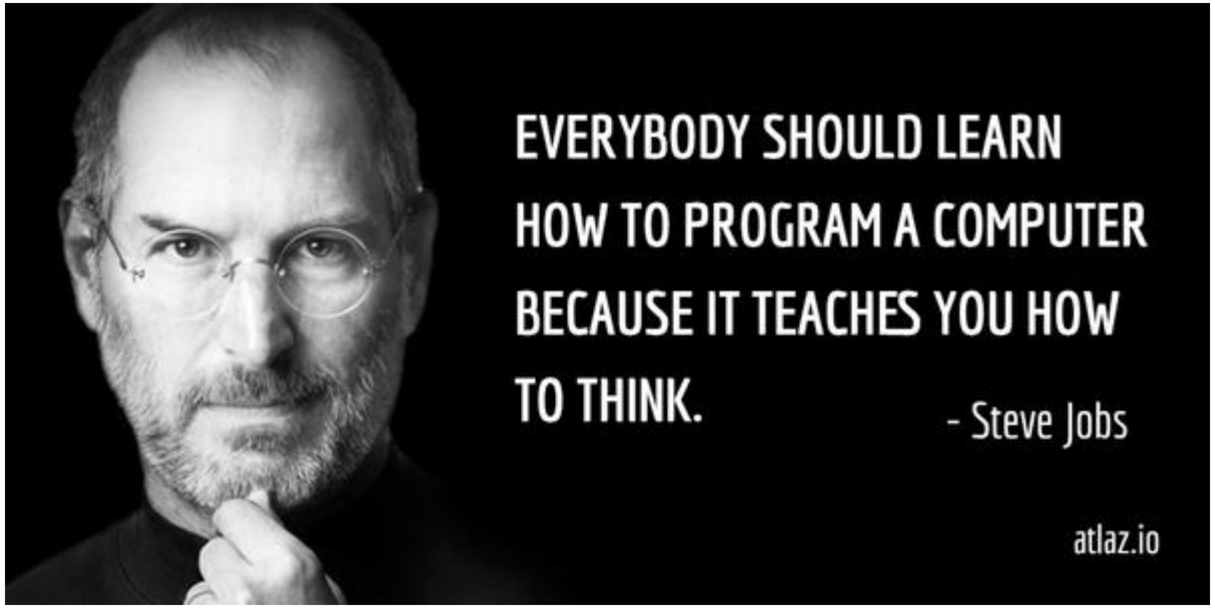
```python
import socket
TCP_IP = '127.0.0.1'
TCP_PORT = 5462
BUFFER_SIZE = 1024
MESSAGE = "Hello, World!"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE.encode('utf-8'))
data = s.recv(BUFFER_SIZE)
s.close()
print("received data:", data)
```

```
received data: b'Moein'
```

Moein

Comments

CHOOSE YOUR WEAPON
C++
JAVA
PYTHON
C



EVERYBODY SHOULD LEARN HOW TO PROGRAM A COMPUTER BECAUSE IT TEACHES YOU HOW TO THINK.

- Steve Jobs

atlaz.io

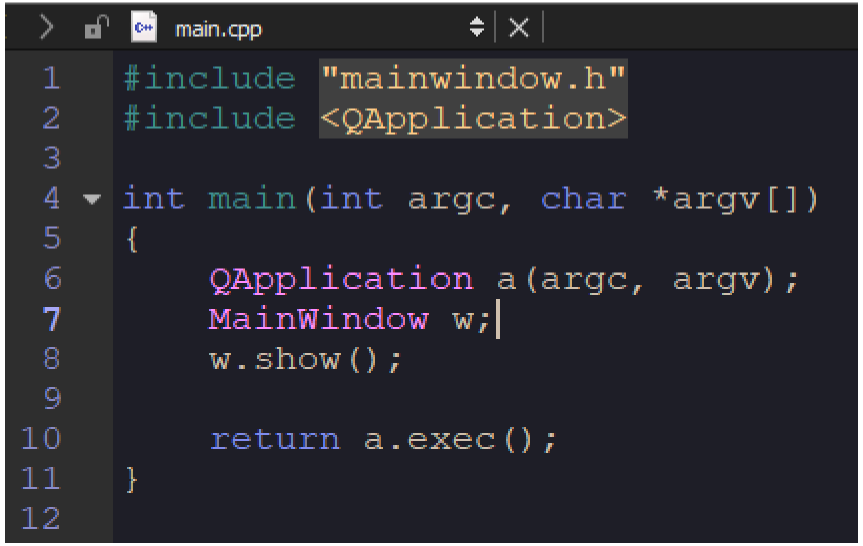
C++

C++

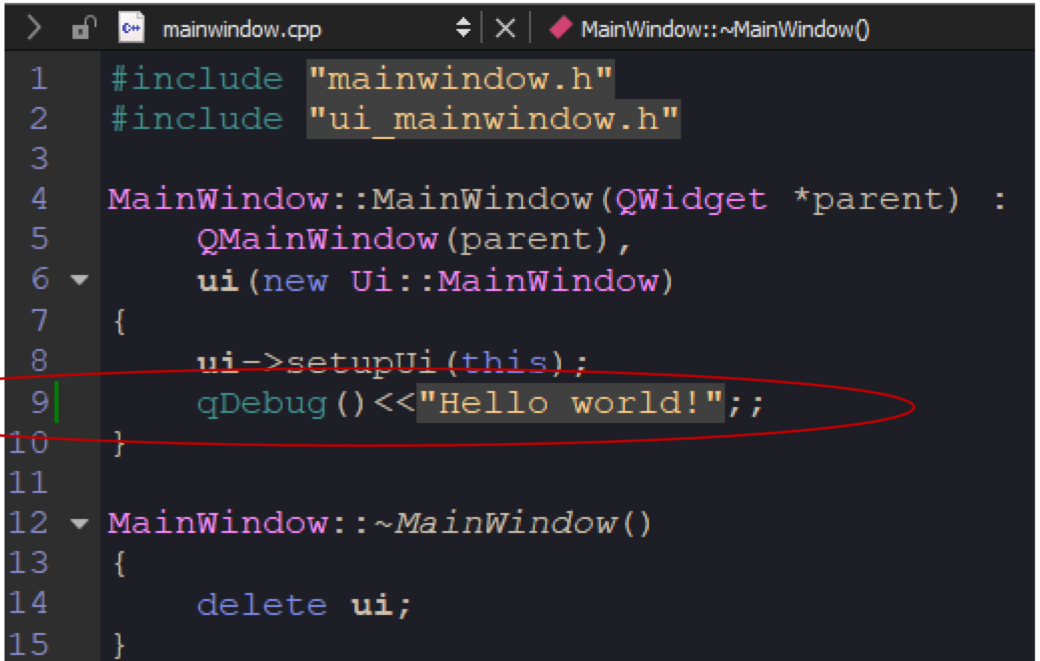
C++ is a general-purpose programming language.
C++ is used to create computer programs. Anything from art applications, music players and even video games!

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

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

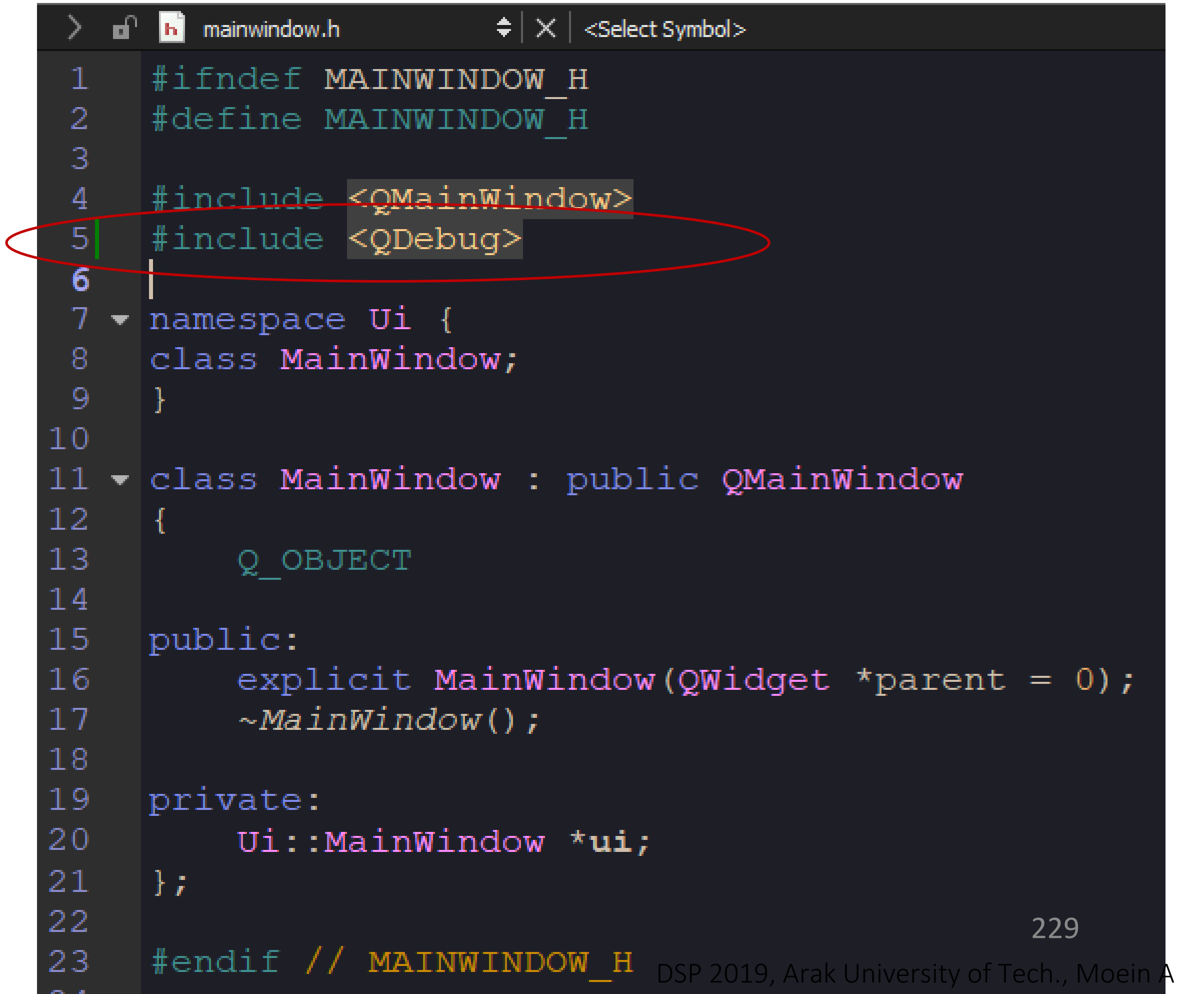
```
int myVariable = 10;


int mark = 90;

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



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

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

    return a.exec();
}

```



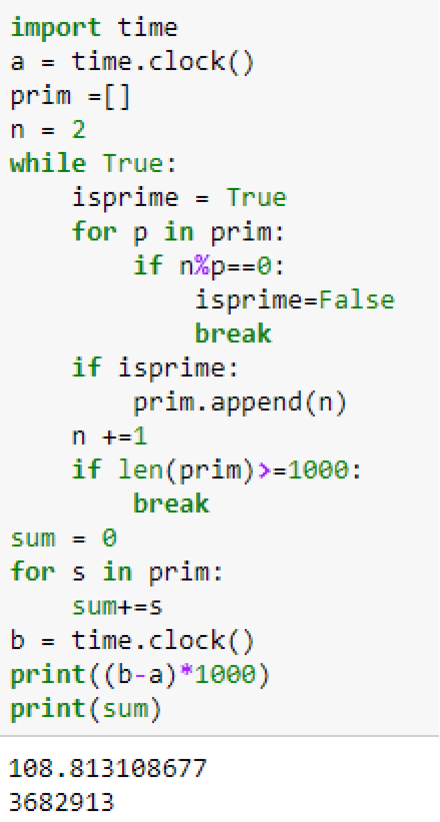
```
#include "mainwindow.h"
#include "ui_mainwindow.h"

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

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



```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QDebug>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

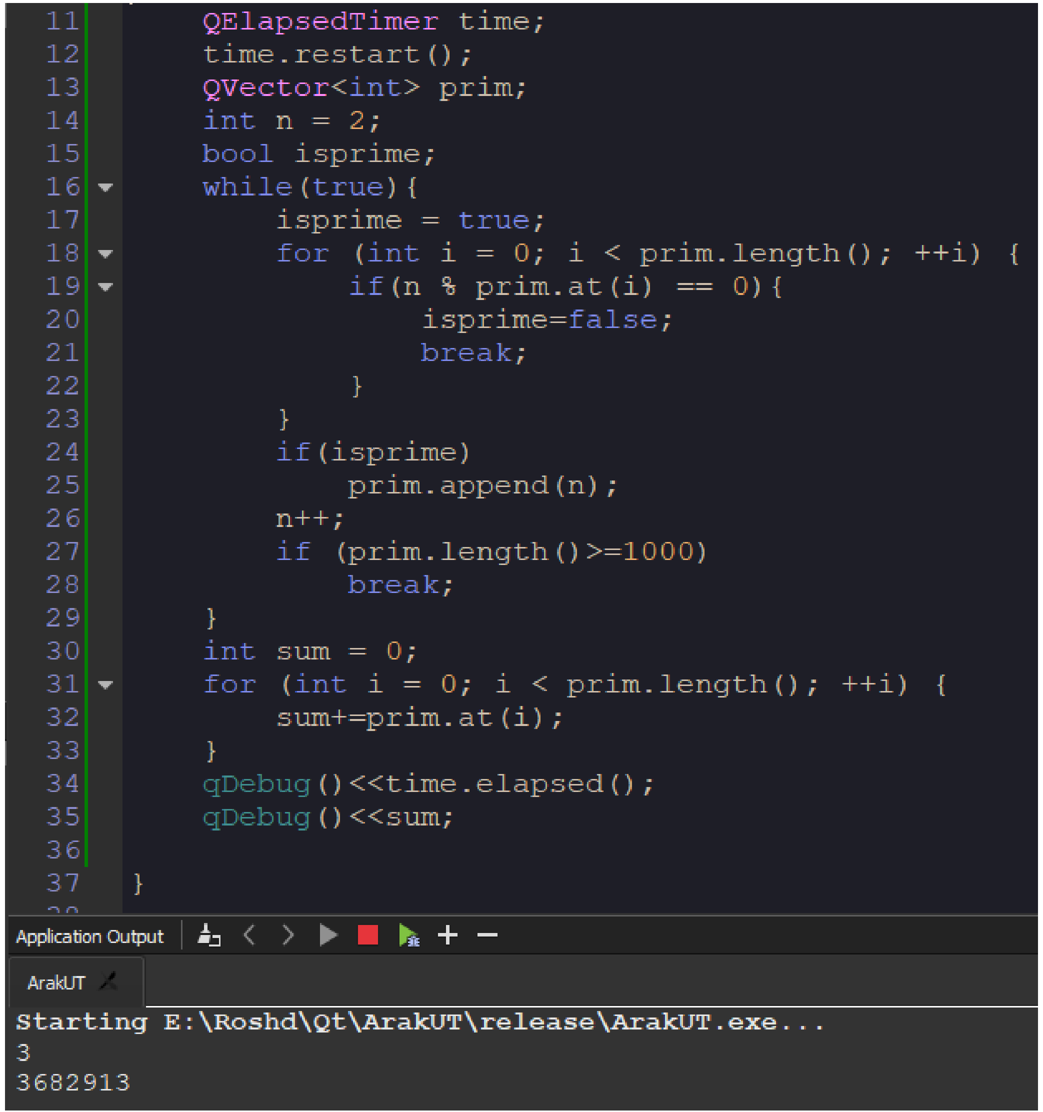
python™ 110 msec

C++ Qt 3 msec



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

```
108.813108677
3682913
```



```cpp
  QElapsedTimer time;
  time.restart();
  QVector<int> prim;
  int n = 2;
  bool isprime;
  while(true){
    isprime = true;
    for (int i = 0; i < prim.length(); ++i) {
      if(n % prim.at(i) == 0){
        isprime=false;
        break;
      }
    }
    if(isprime)
      prim.append(n);
    n++;
    if (prim.length()>=1000)
      break;
  }
  int sum = 0;
  for (int i = 0; i < prim.length(); ++i) {
    sum+=prim.at(i);
  }
  qDebug()<<time.elapsed();
  qDebug()<<sum;

}
```

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

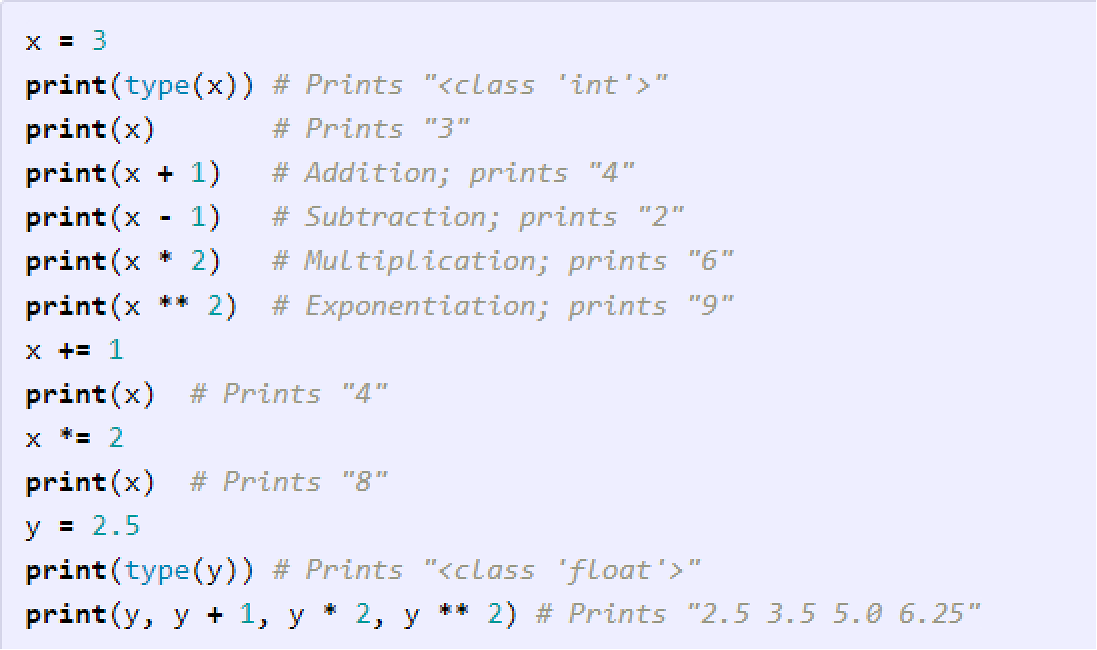
Python for DSP
Numpy

python™

Basic data types:

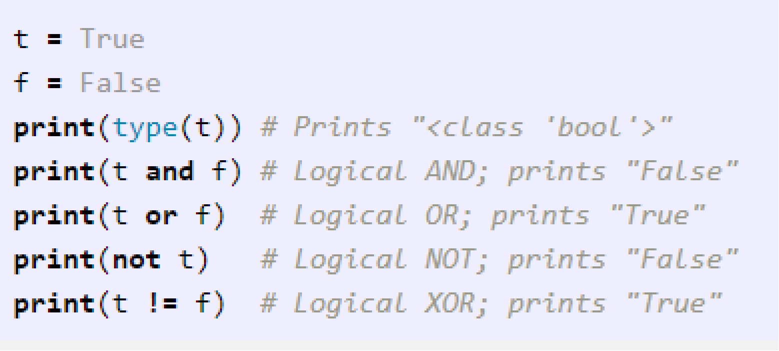
Numbers, Booleans, Strings

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



```python
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.):

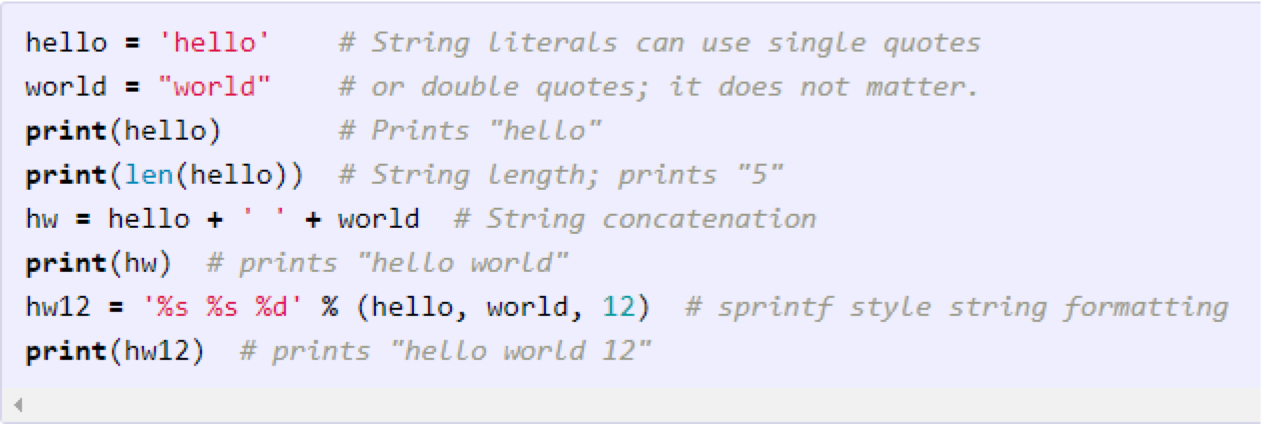


```python
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"
print(t or f)  # Logical OR; prints "True"
print(not t)   # Logical NOT; prints "False"
print(t != f)  # Logical XOR; prints "True"
```

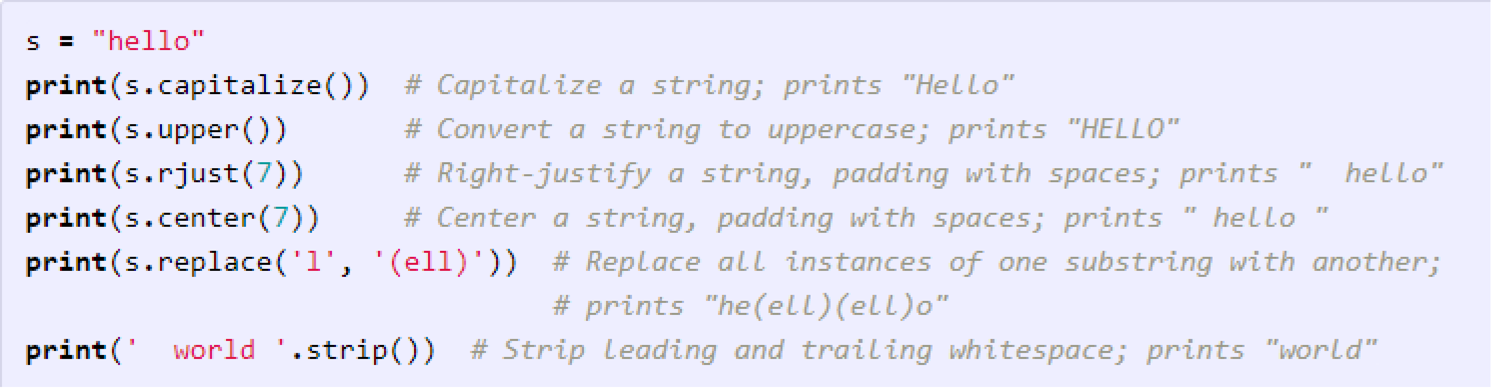
Basic data types:

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"
```

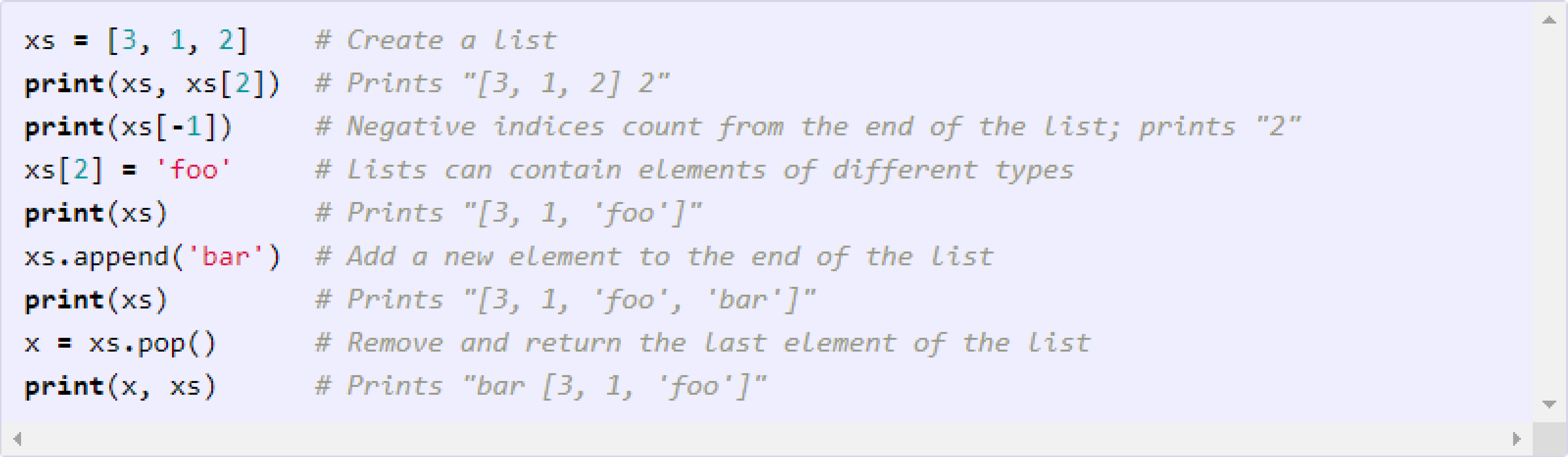


```
s = "hello"
print(s.capitalize())  # Capitalize a string; prints "Hello"
print(s.upper())       # Convert a string to uppercase; prints "HELLO"
print(s.rjust(7))      # Right-justify a string, padding with spaces; prints "  hello"
print(s.center(7))     # Center a string, padding with spaces; prints " 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"
```

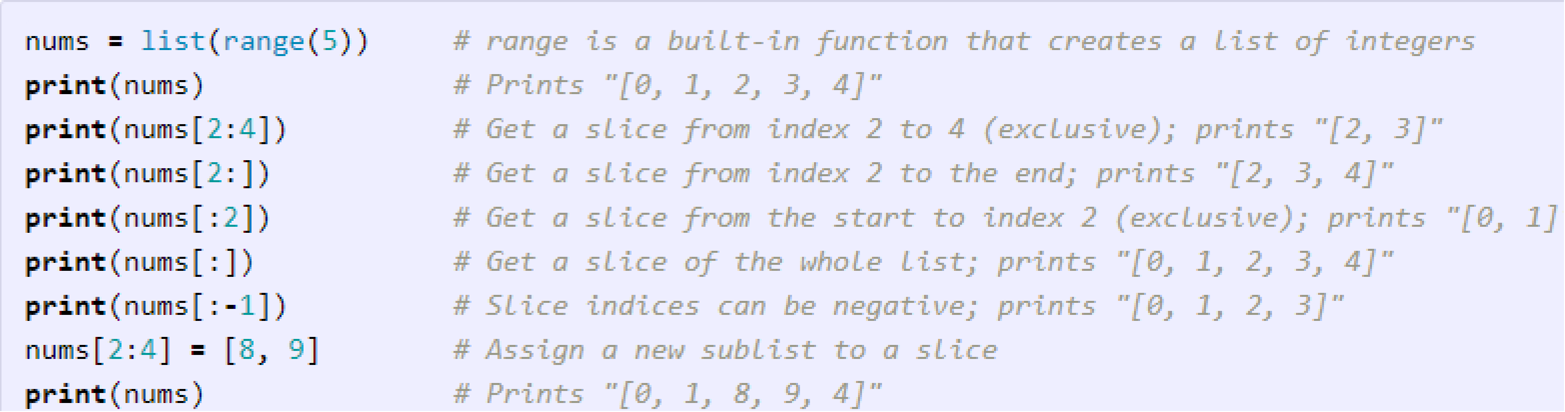
python™

Containers: List, Dictionary, Set, Tuple

List



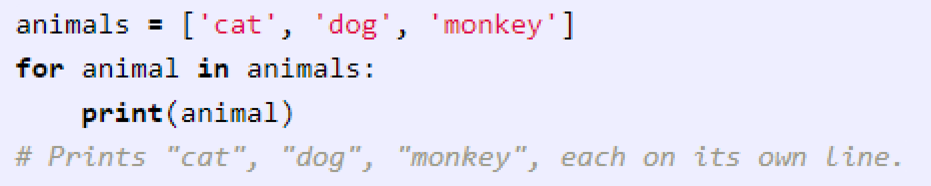
```
xs = [3, 1, 2]    # Create a list
print(xs, xs[2])  # Prints "[3, 1, 2] 2"
print(xs[-1])     # Negative indices count from the end of the list; prints "2"
xs[2] = 'foo'     # Lists can contain elements of different types
print(xs)         # Prints "[3, 1, 'foo']"
xs.append('bar')  # Add a new element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



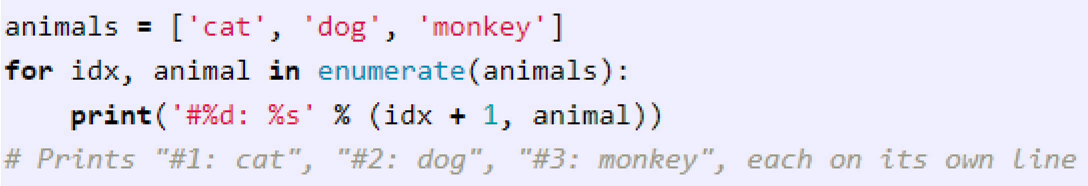
```
nums = list(range(5))     # range is a built-in function that creates a list of integers
print(nums)               # Prints "[0, 1, 2, 3, 4]"
print(nums[2:4])          # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print(nums[2:])           # Get a slice from index 2 to the end; prints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

Containers: List, Dictionary, Set, Tuple

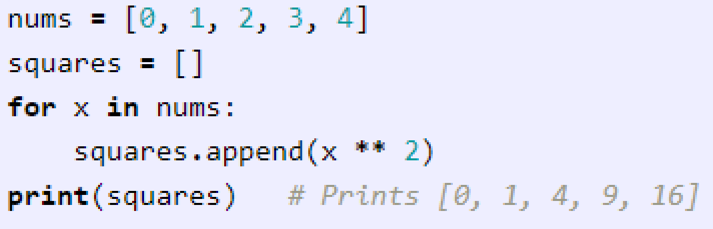
List



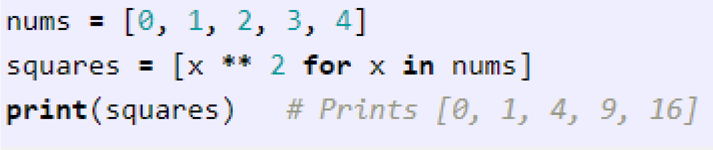
```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
# Prints "cat", "dog", "monkey", each on its own line.
```



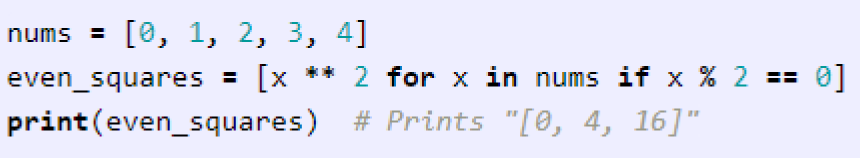
```
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# Prints "#1: cat", "#2: dog", "#3: monkey", each on its own line
```



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



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

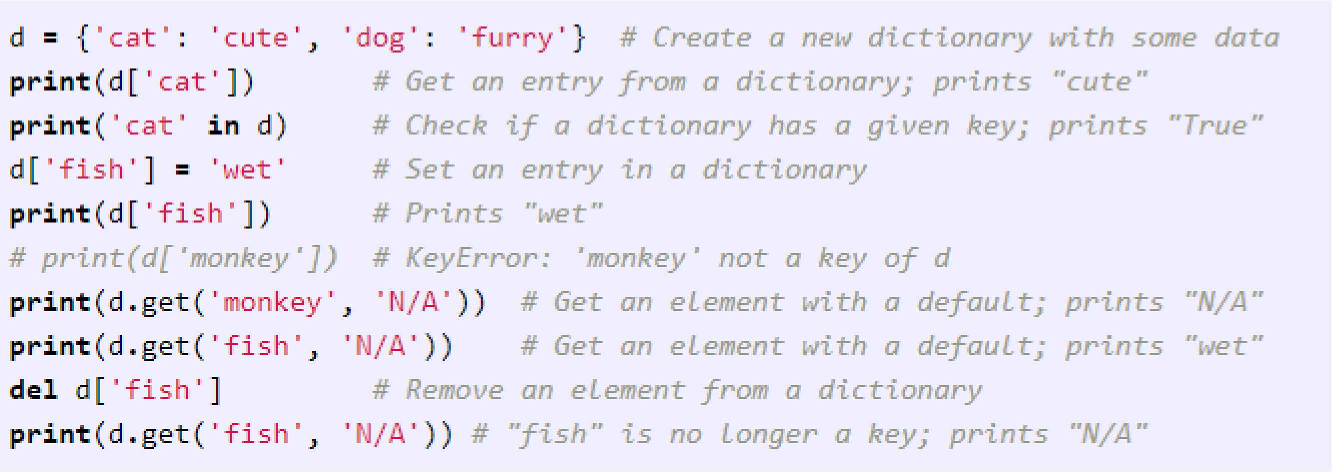


```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)  # Prints "[0, 4, 16]"
```

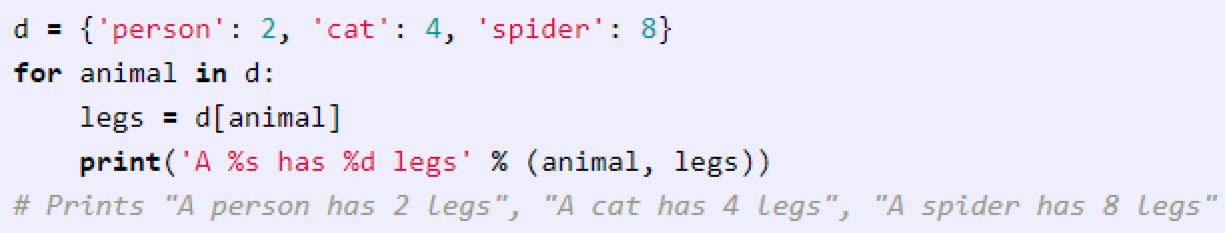
python™

Containers: List, Dictionary, Set, Tuple

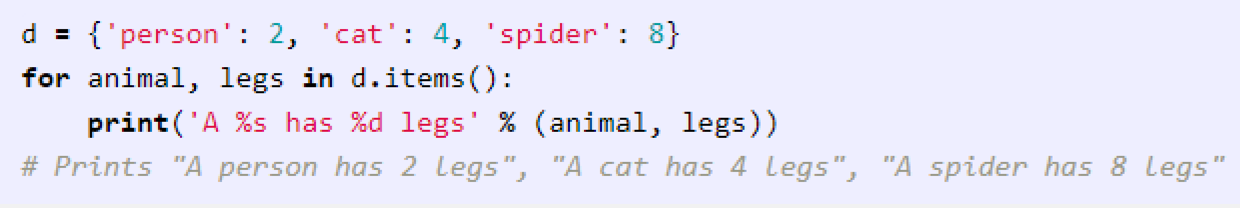
Dictionary



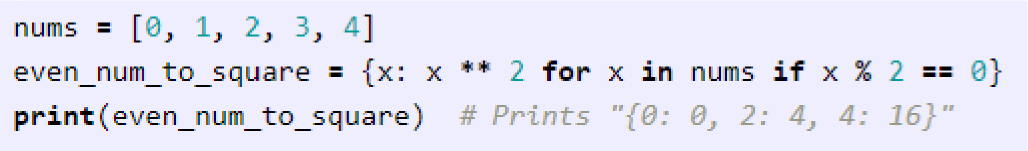
```
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet'     # Set an entry in a dictionary
print(d['fish'])  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



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



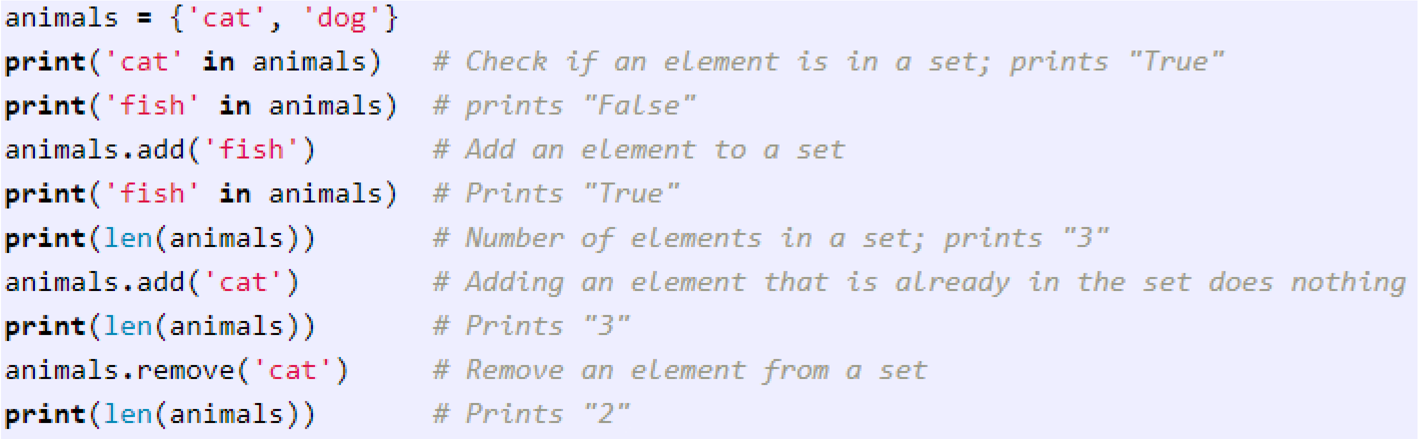
```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
```



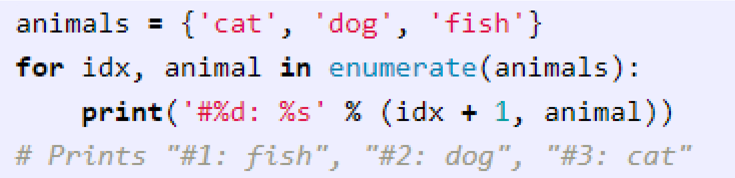
```
nums = [0, 1, 2, 3, 4]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print(even_num_to_square)  # Prints "{0: 0, 2: 4, 4: 16}"
```

Containers: List, Dictionary, Set, Tuple

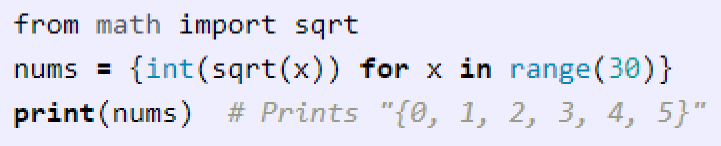
Set



```
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"
```

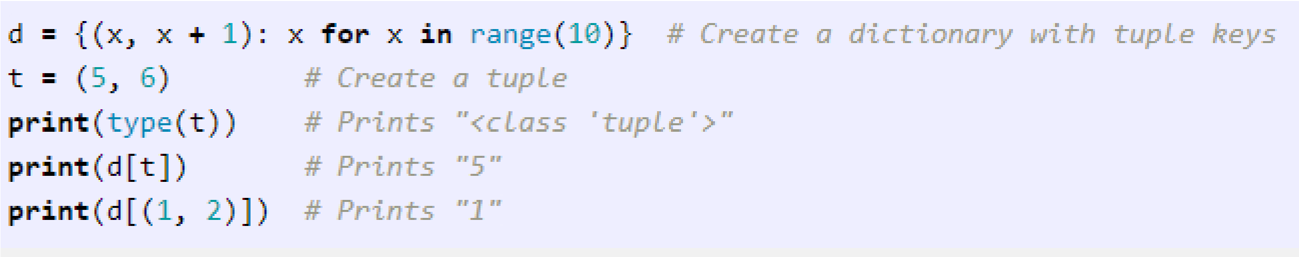


```
from math import sqrt
nums = {int(sqrt(x)) for x in range(30)}
print(nums)  # Prints "{0, 1, 2, 3, 4, 5}"
```

Containers: List, Dictionary, Set, Tuple

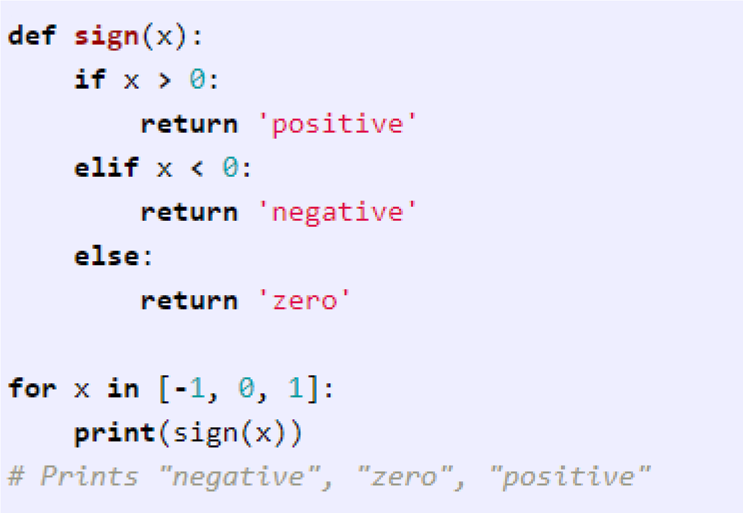
Tuple

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.

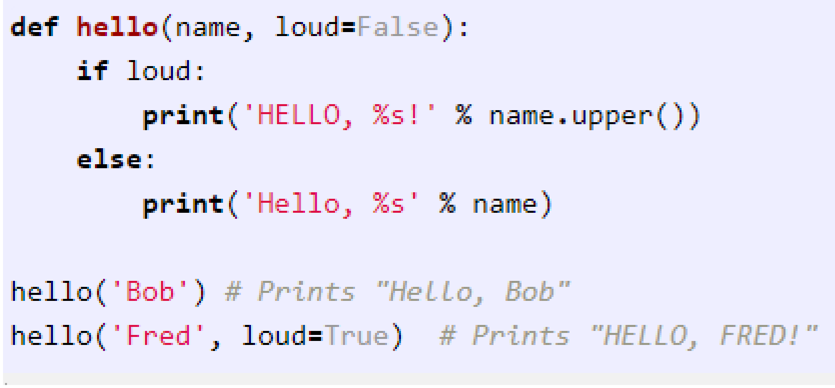


```
d = {(x, x + 1): x for x in range(10)}  # Create a dictionary with tuple keys
t = (5, 6)        # Create a tuple
print(type(t))    # Prints "<class 'tuple'>"
print(d[t])       # Prints "5"
print(d[(1, 2)])  # Prints "1"
```

Functions

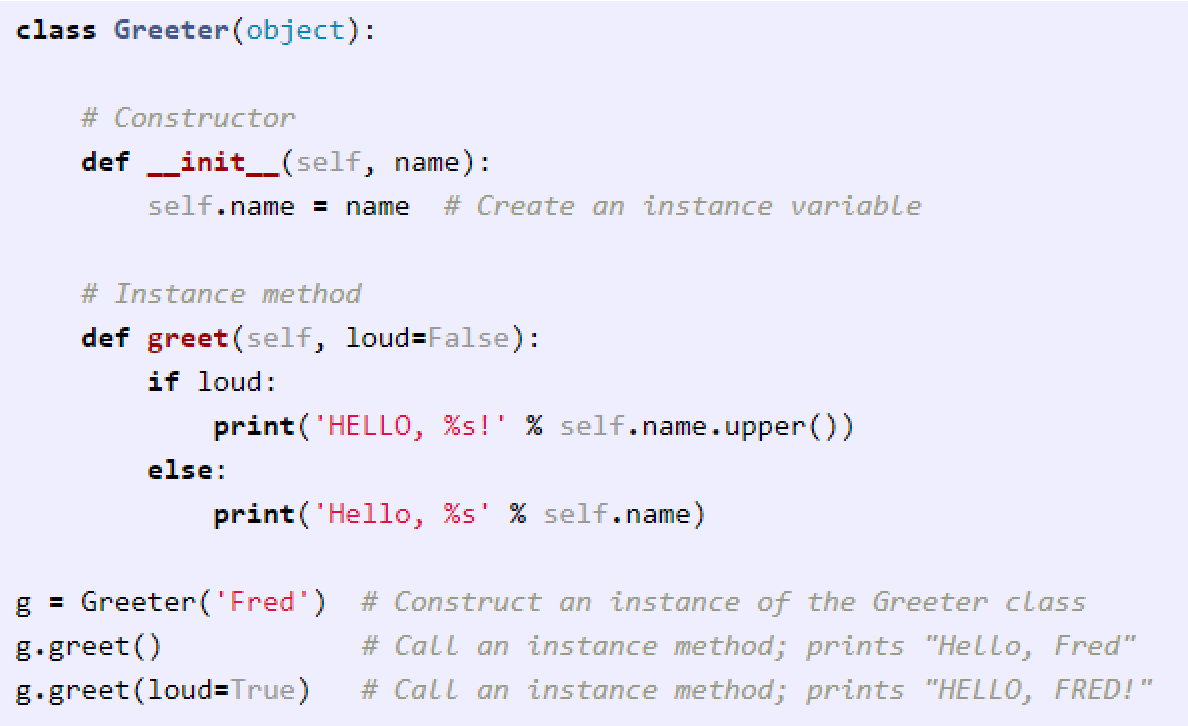


```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

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

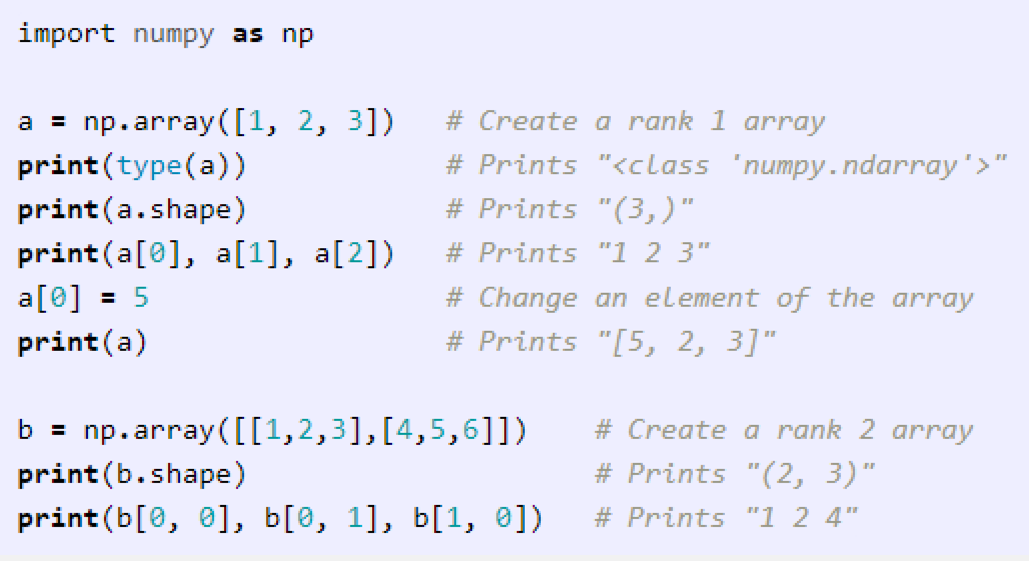


```python
class Greeter(object):

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

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

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

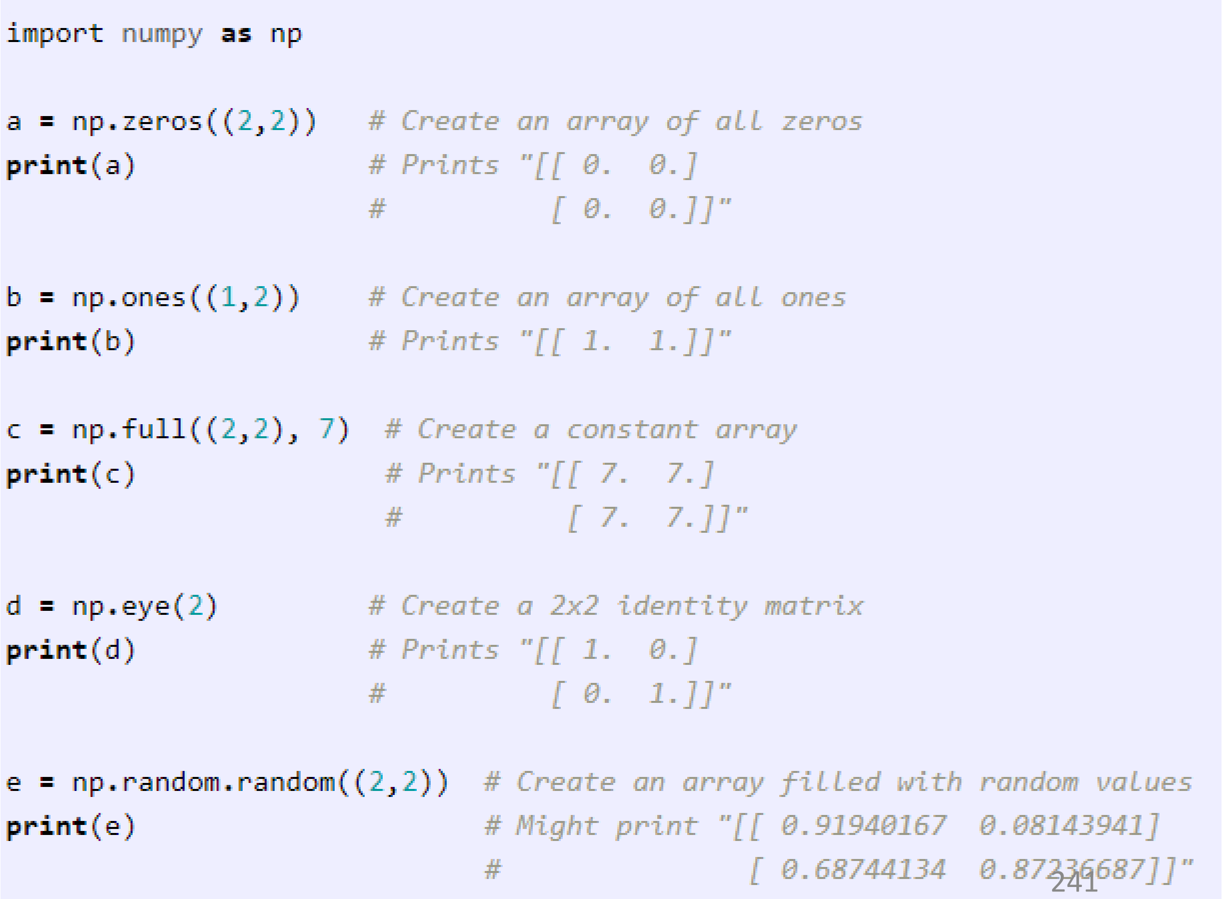
python™

libraries Numpy SciPy Matplotlib

Numpy Arrays



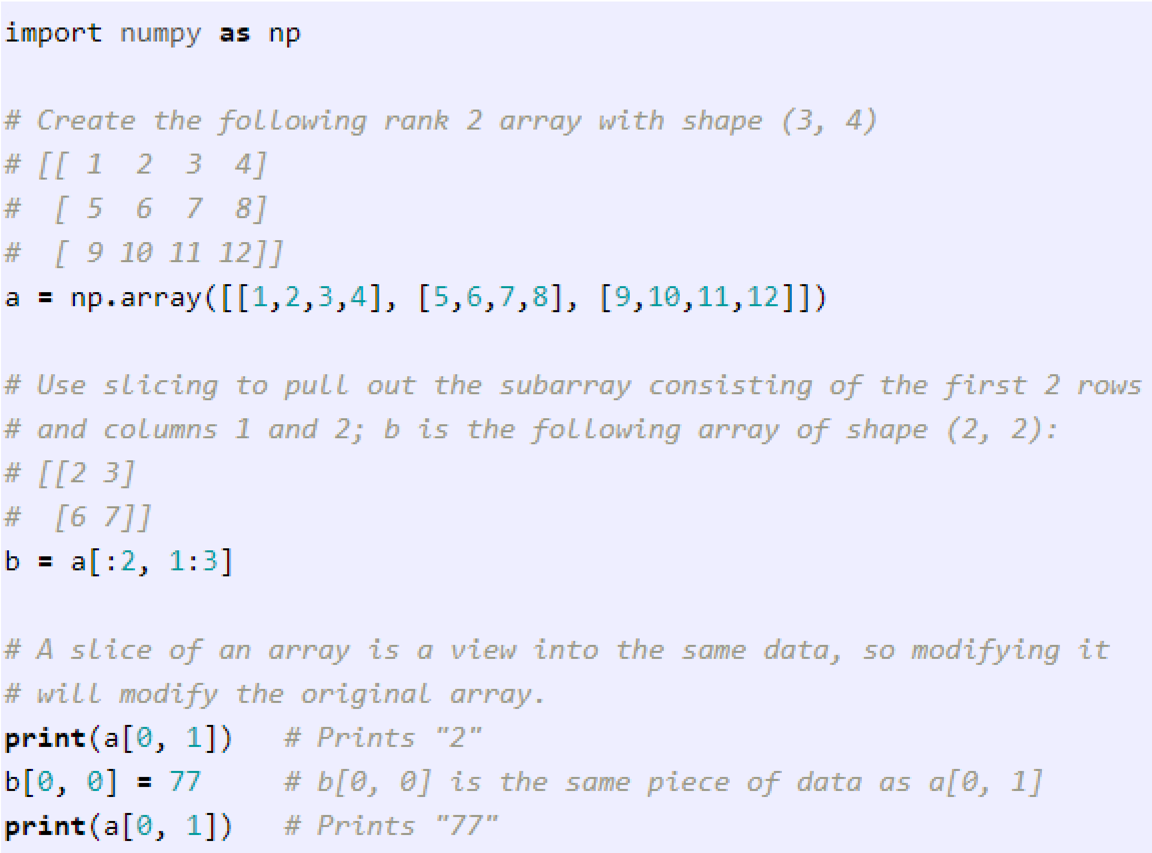
```
import numpy as np

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

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



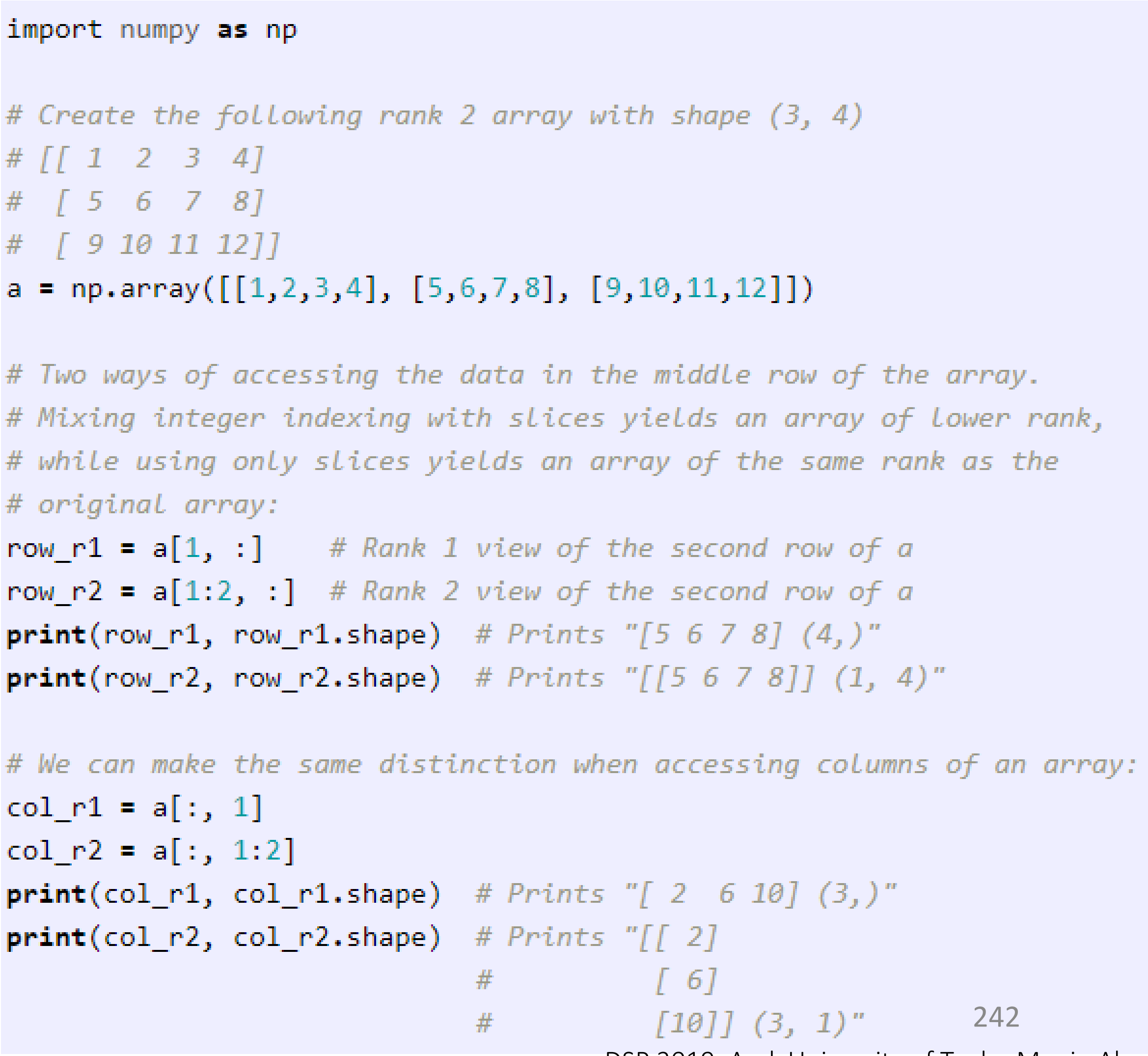
```
import numpy as np

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

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

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

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

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

libraries Numpy SciPy Matplotlib

Numpy Array indexing



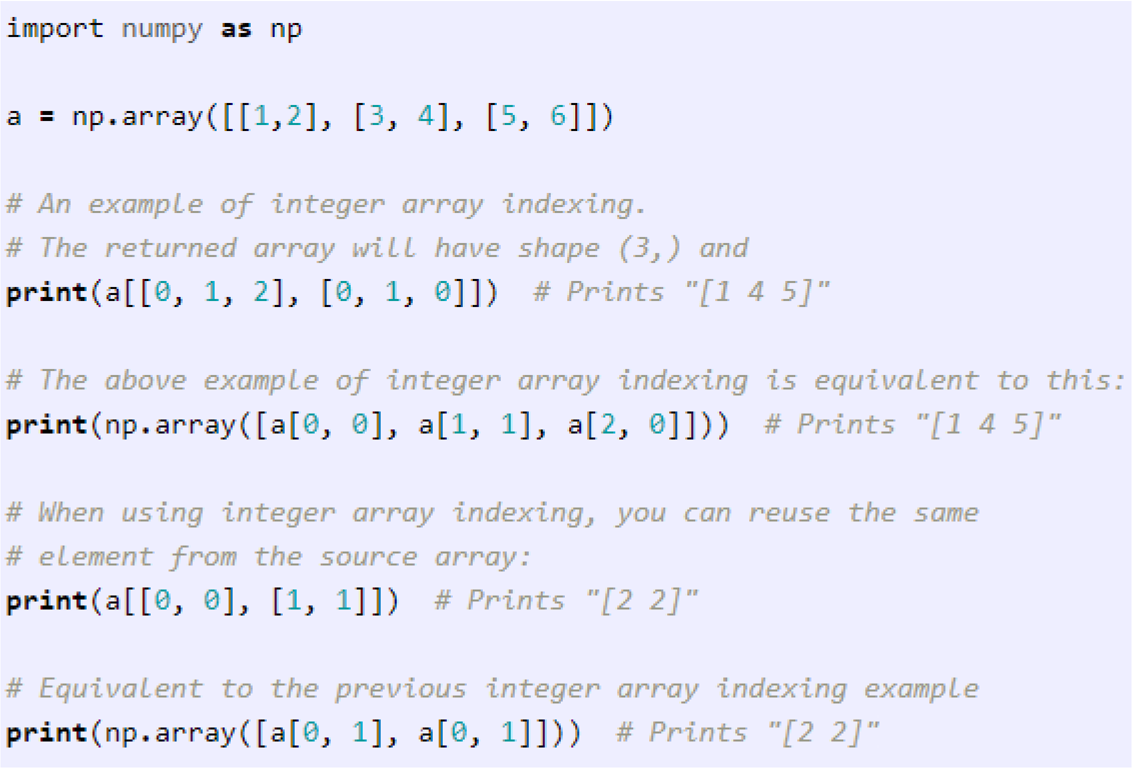
```
import numpy as np

# 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"
```



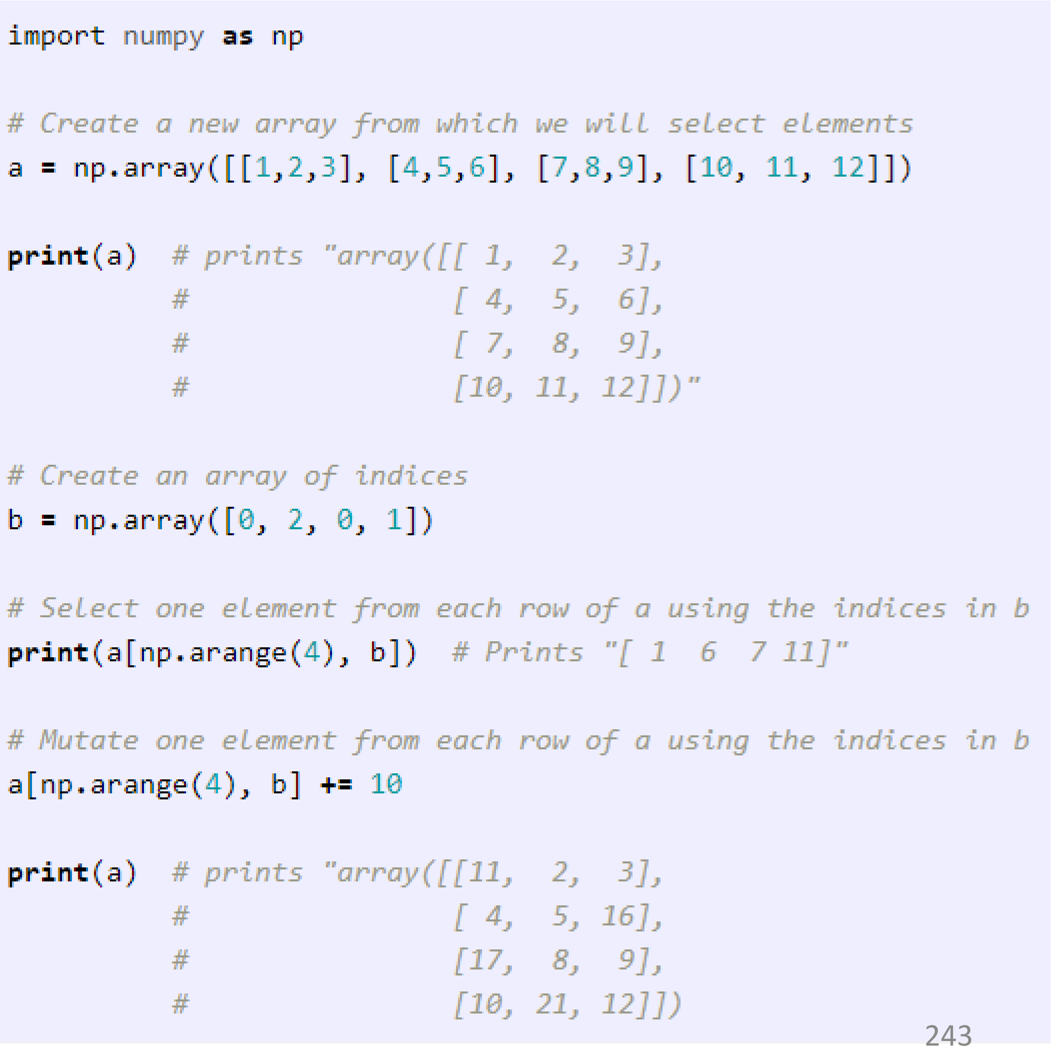
```
import numpy as np

# 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



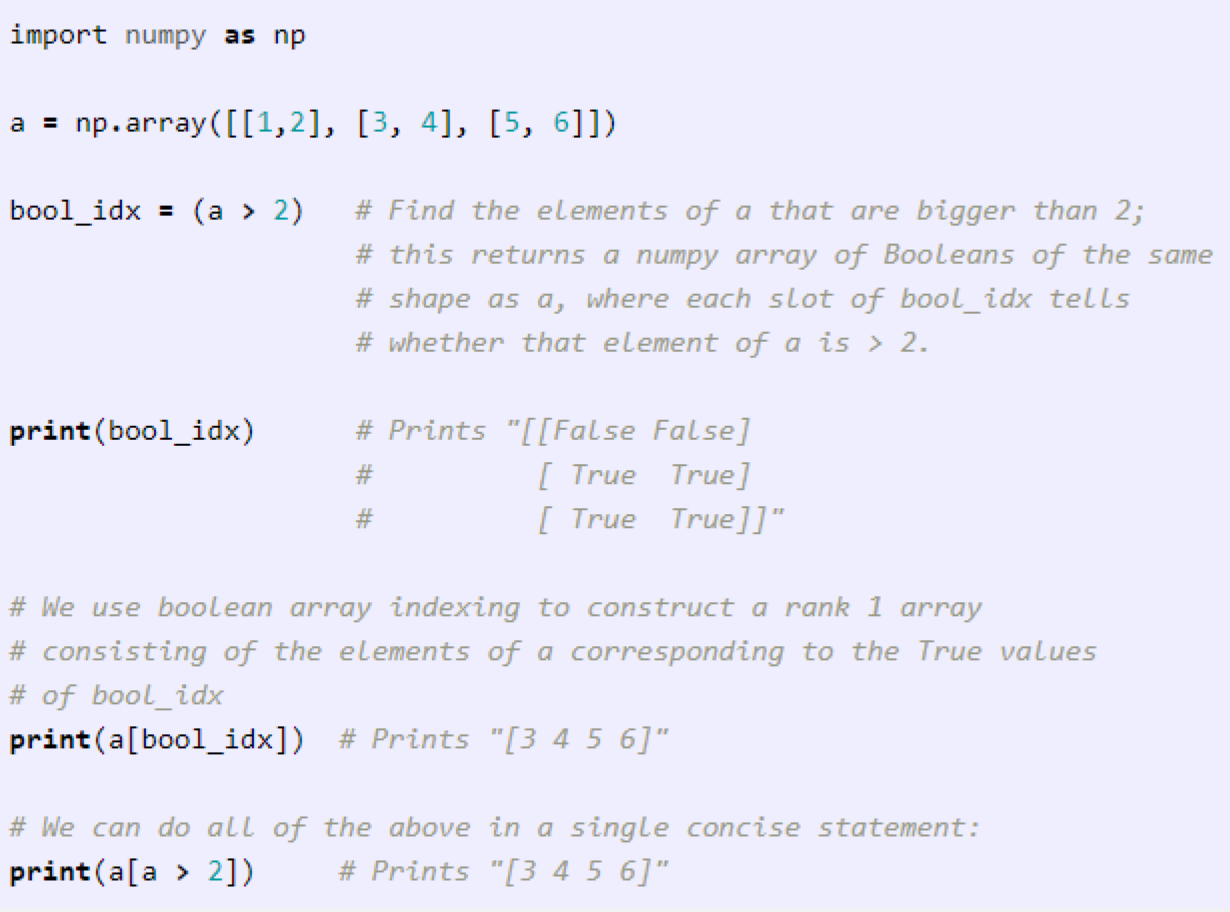
```
import numpy as np

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

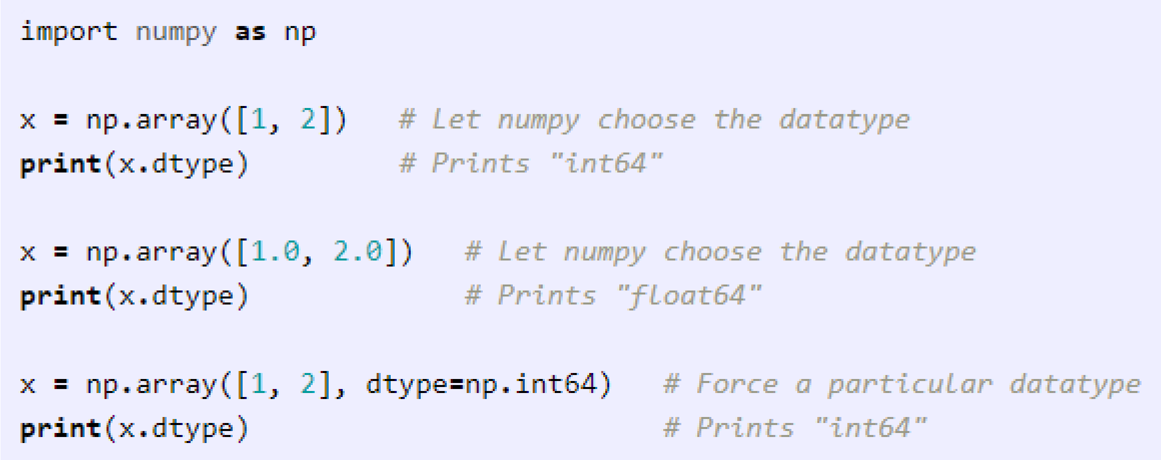
Numpy Array indexing



```
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

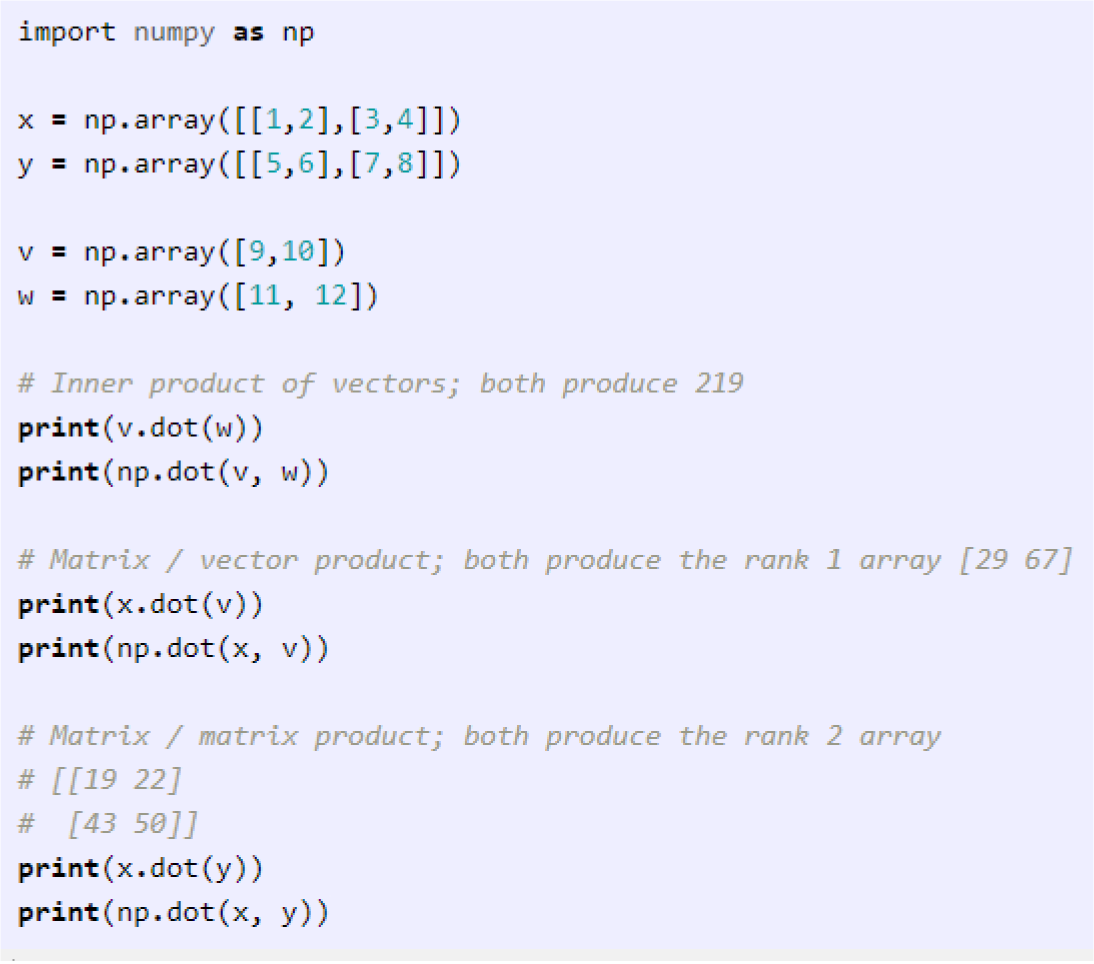


```
import numpy as np

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"
```

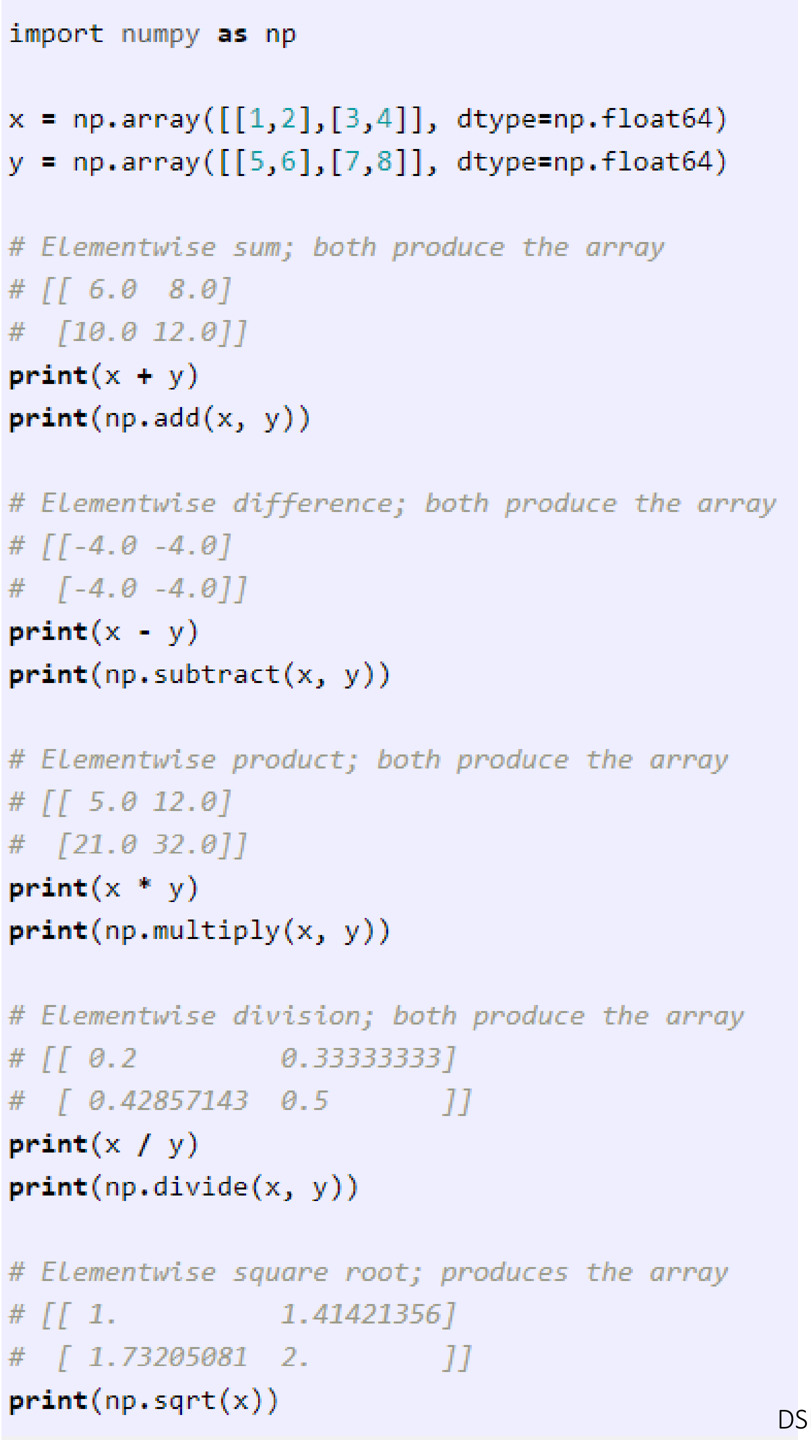
python™

libraries Numpy SciPy Matplotlib

Numpy Array math



```
import numpy as np

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

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

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

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

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

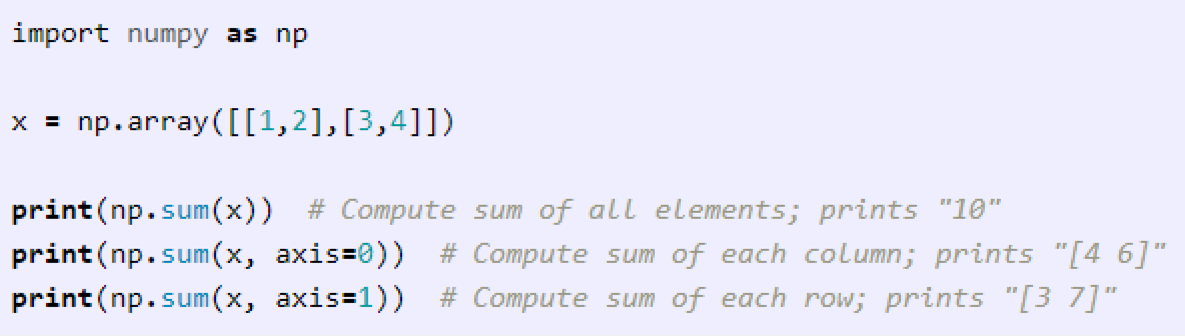


```
import numpy as np

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

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

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

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

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

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

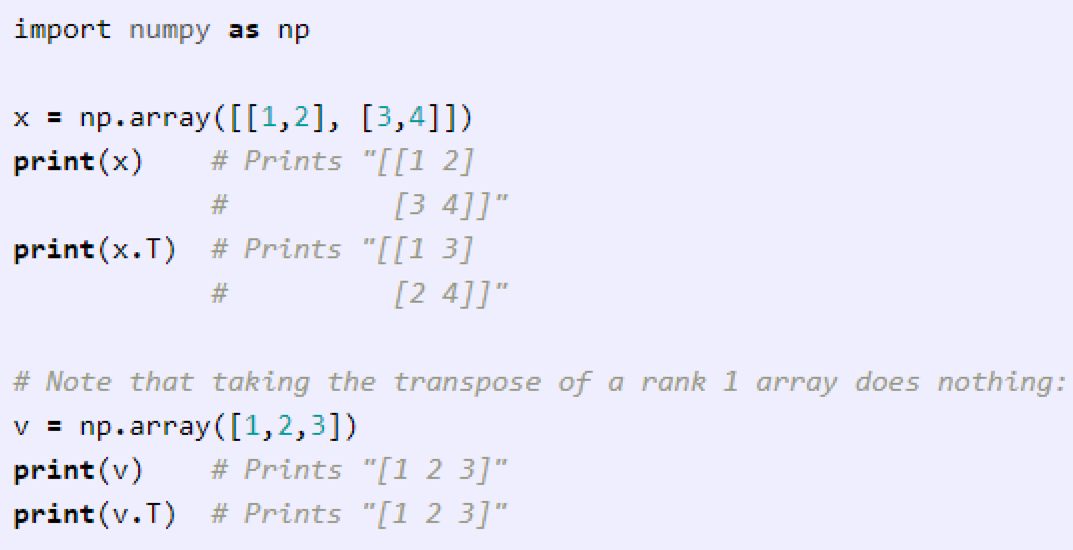
python™

libraries Numpy SciPy Matplotlib

Numpy Array math



```python
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]"
```

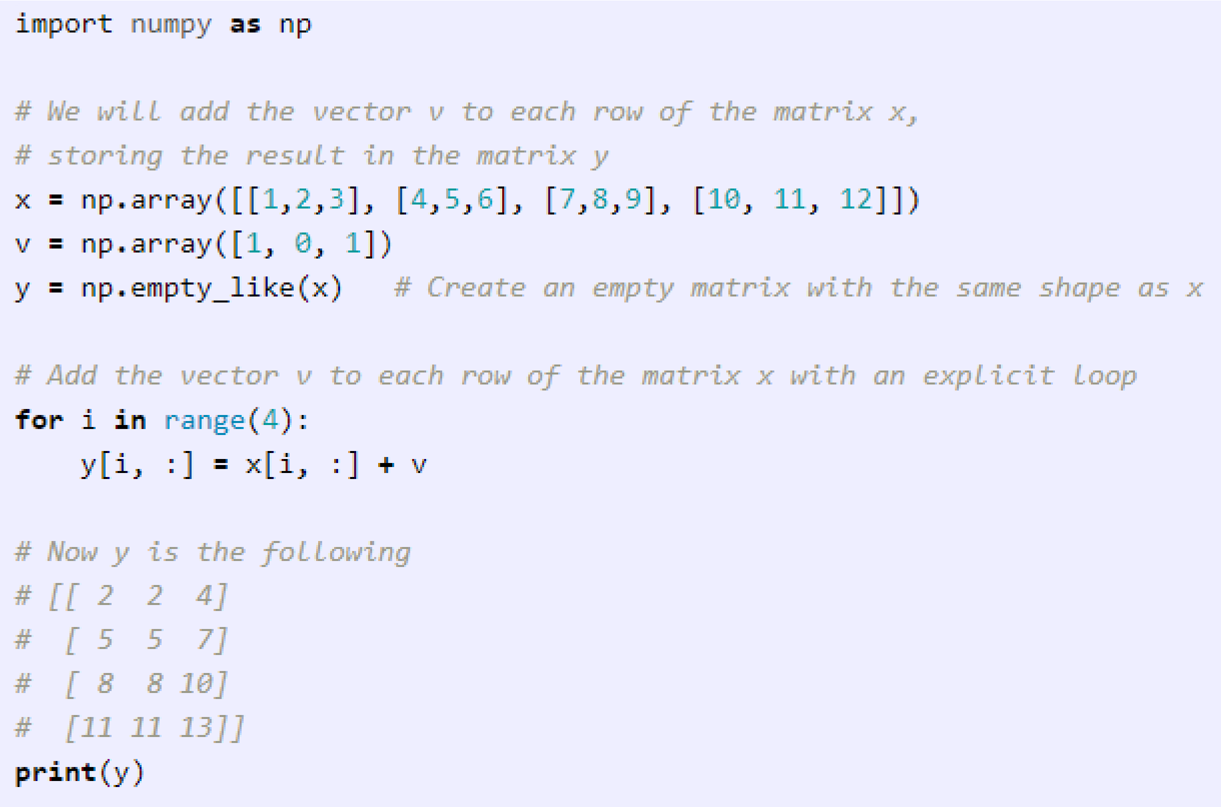


```python
import numpy as np

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]"
```

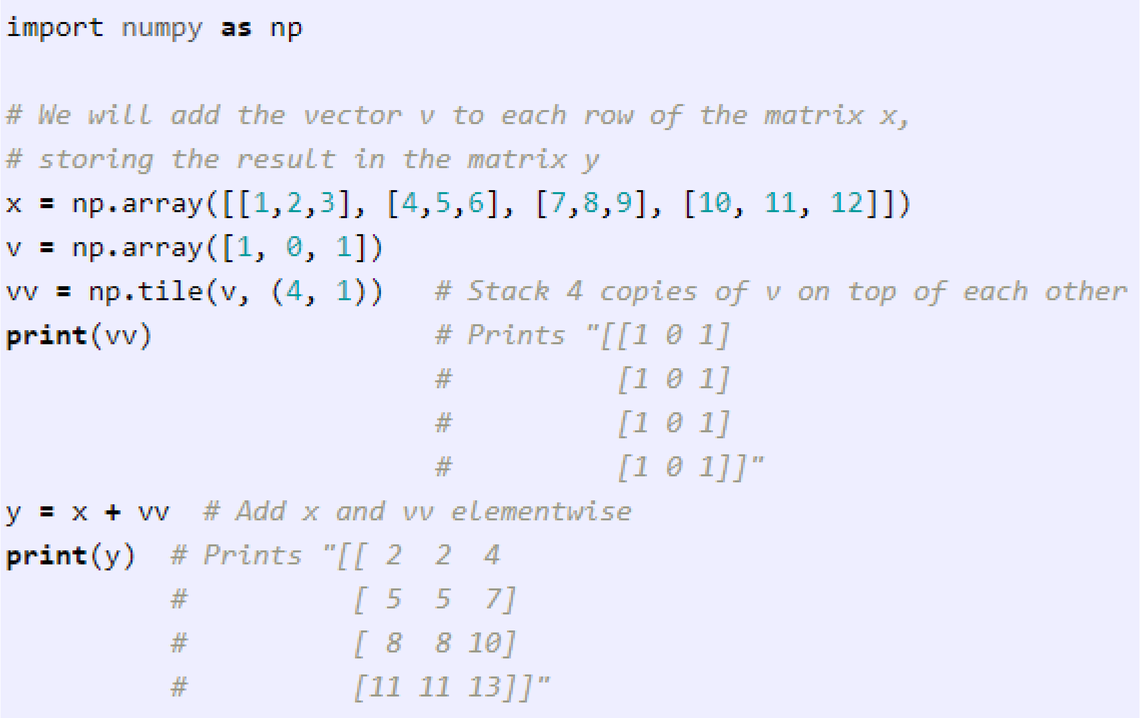
python™

libraries Numpy SciPy Matplotlib

Numpy Broadcasting



```
import numpy as np

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

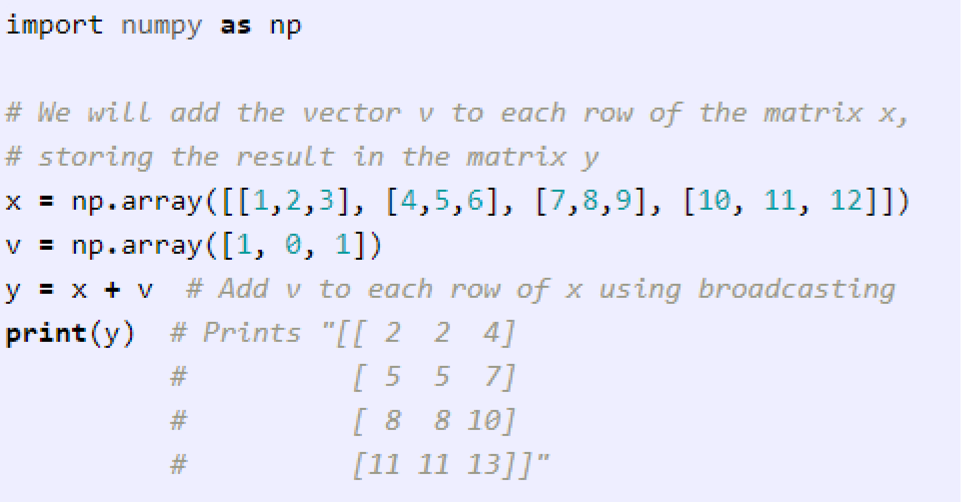


```
import numpy as np

# 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]]"
```

python™

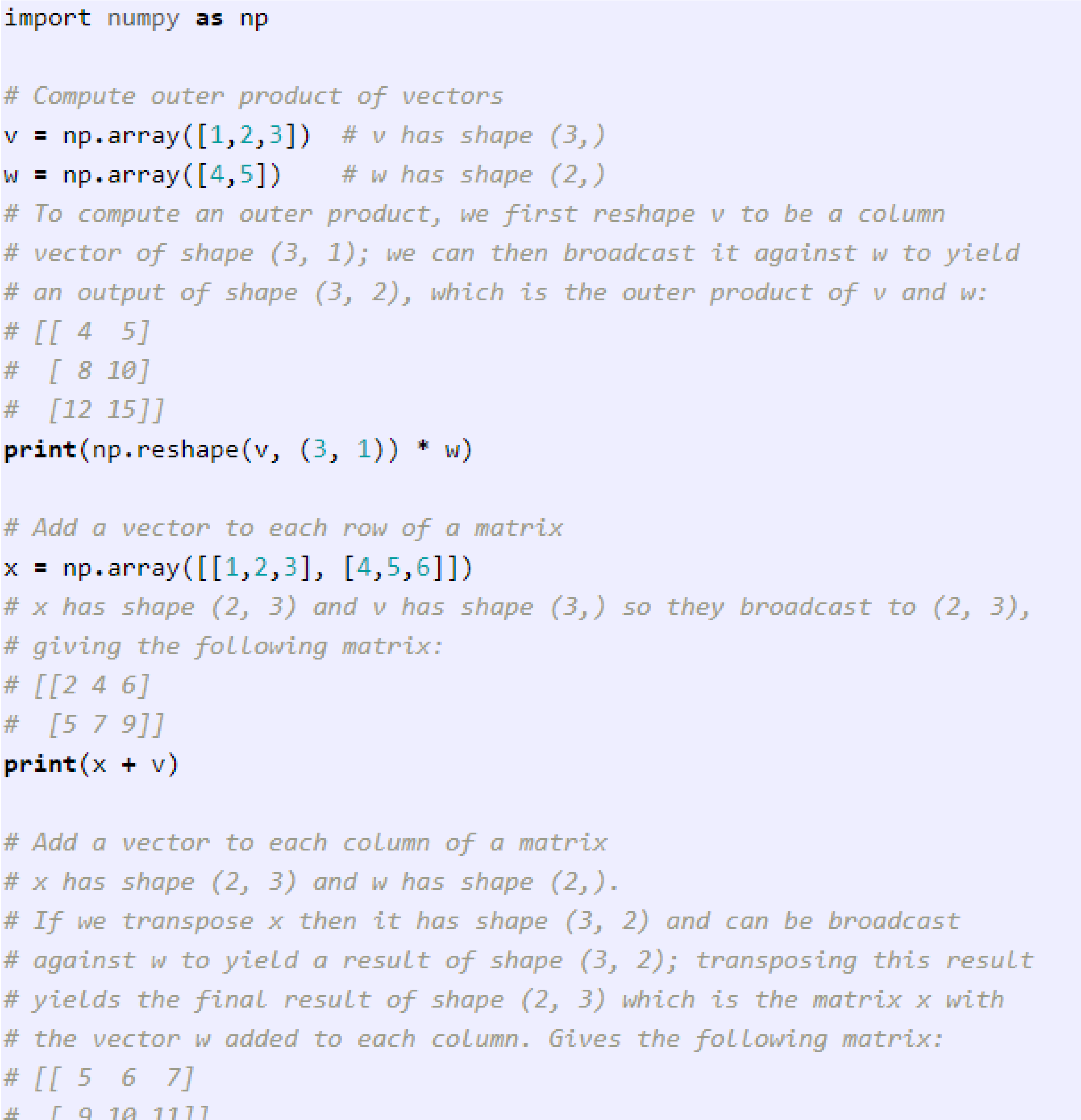
libraries Numpy SciPy Matplotlib

Numpy Broadcasting

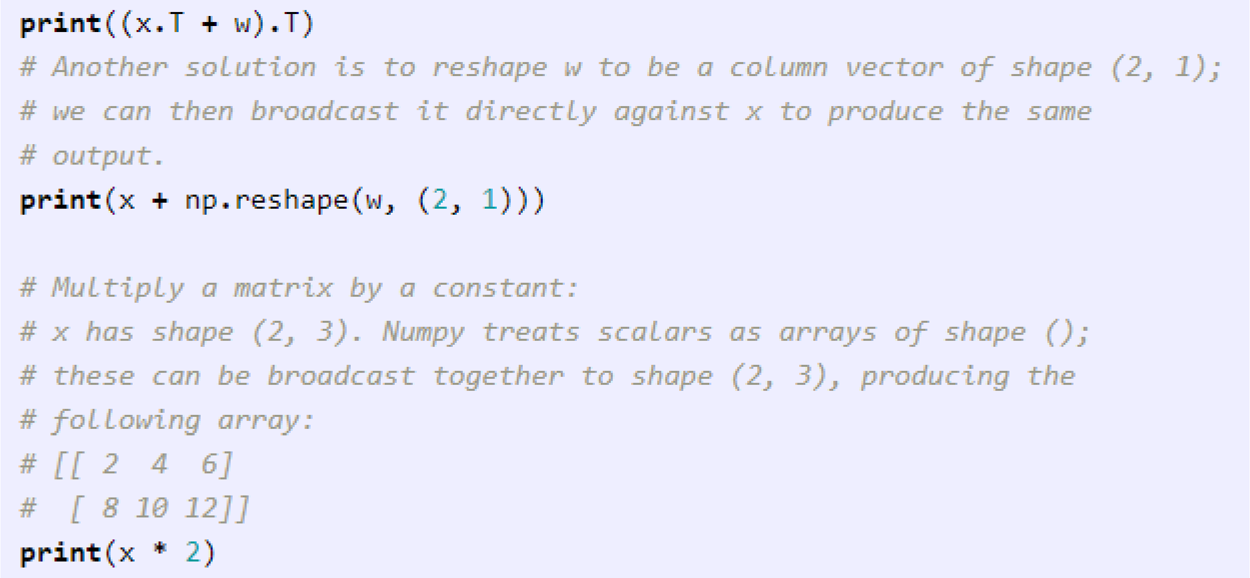


```
import numpy as np

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

Numpy Broadcasting



```
import numpy as np

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

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

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

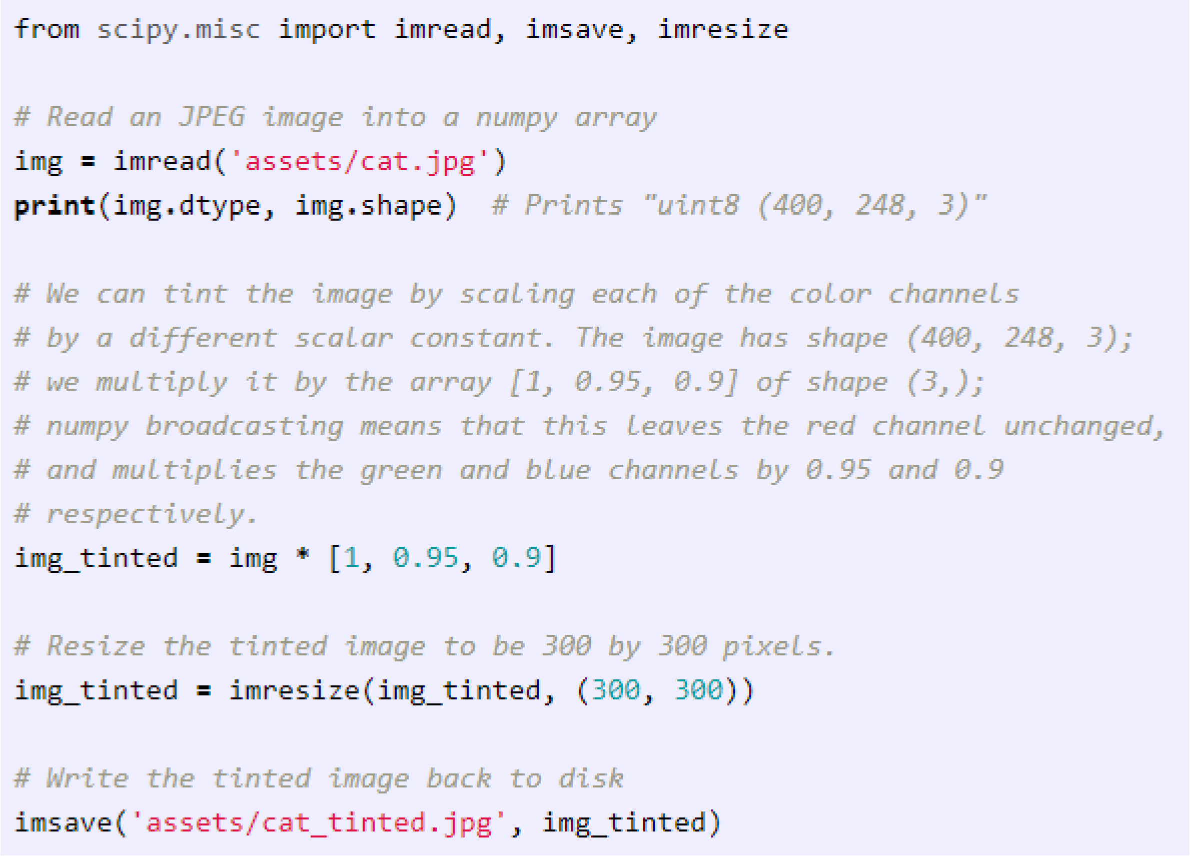


```
print((x.T + w).T)
# Another solution is to reshape w to be a column vector of shape (2, 1);
# we can then broadcast it directly against x to produce the same
# output.
print(x + np.reshape(w, (2, 1)))

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

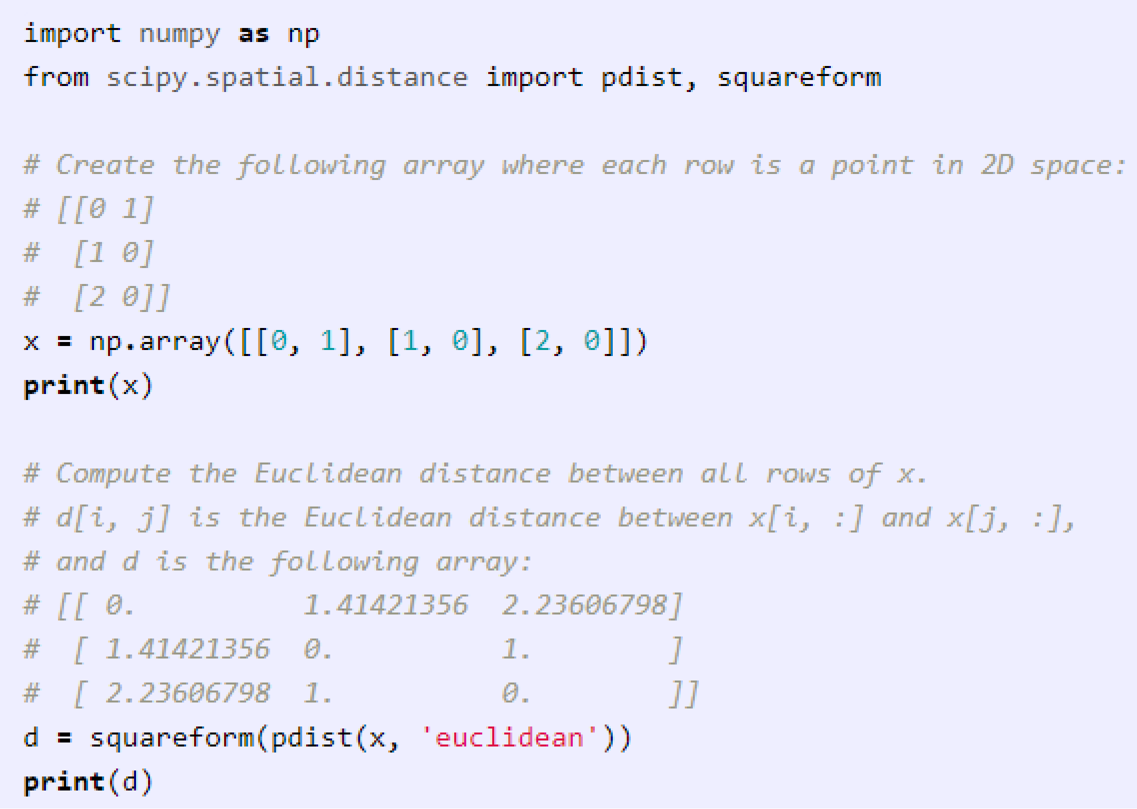
python™

libraries Numpy SciPy Matplotlib

SciPy



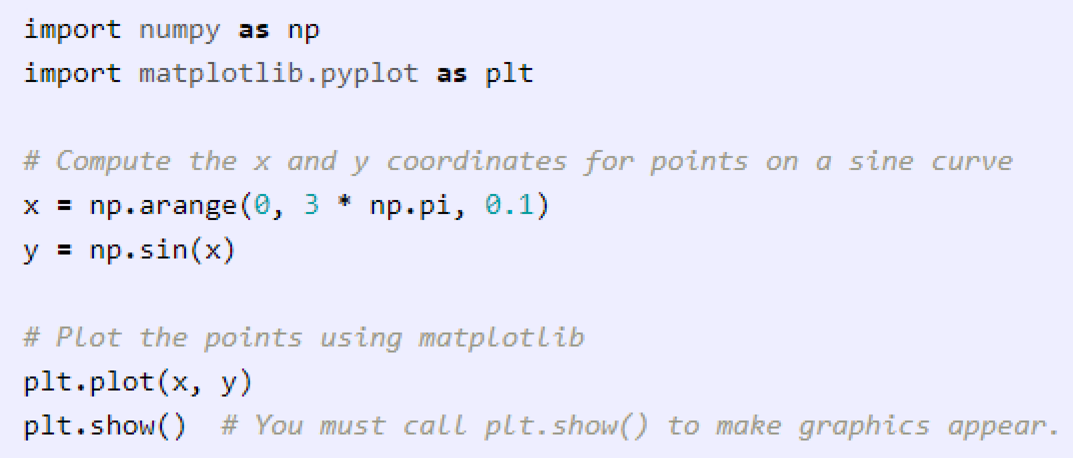
```
from scipy.misc import imread, imsave, imresize

# 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)
```



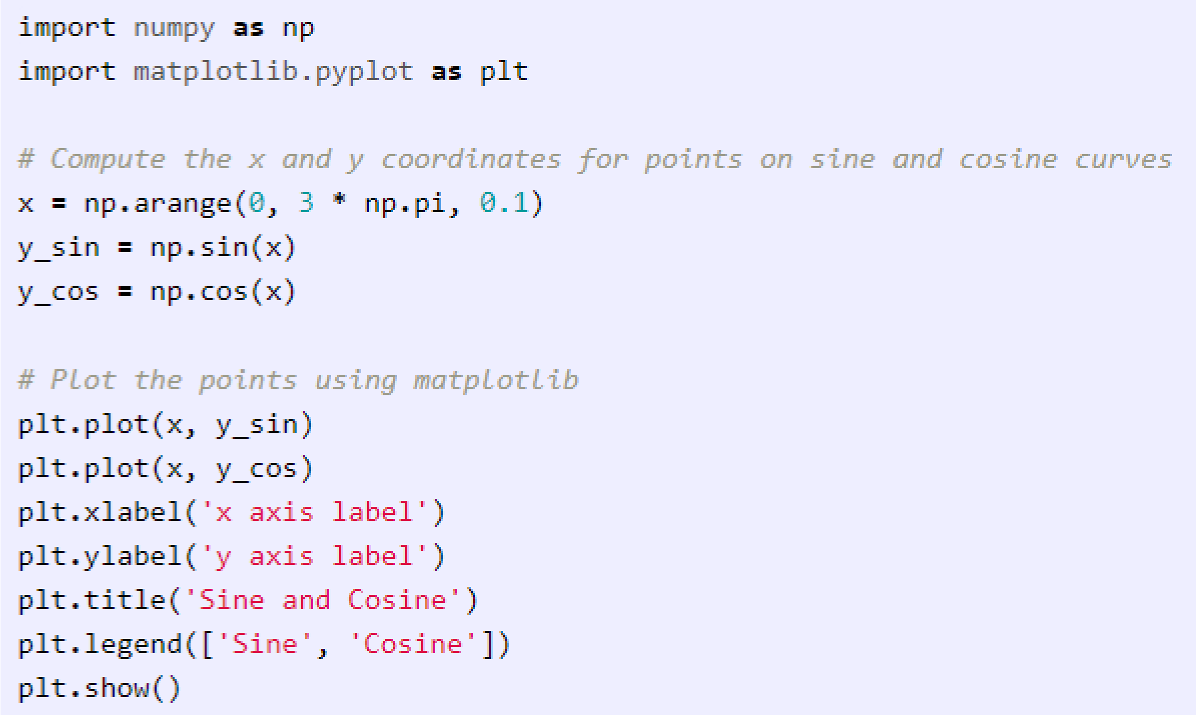
```
import numpy as np
from scipy.spatial.distance import pdist, squareform

# 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.
```

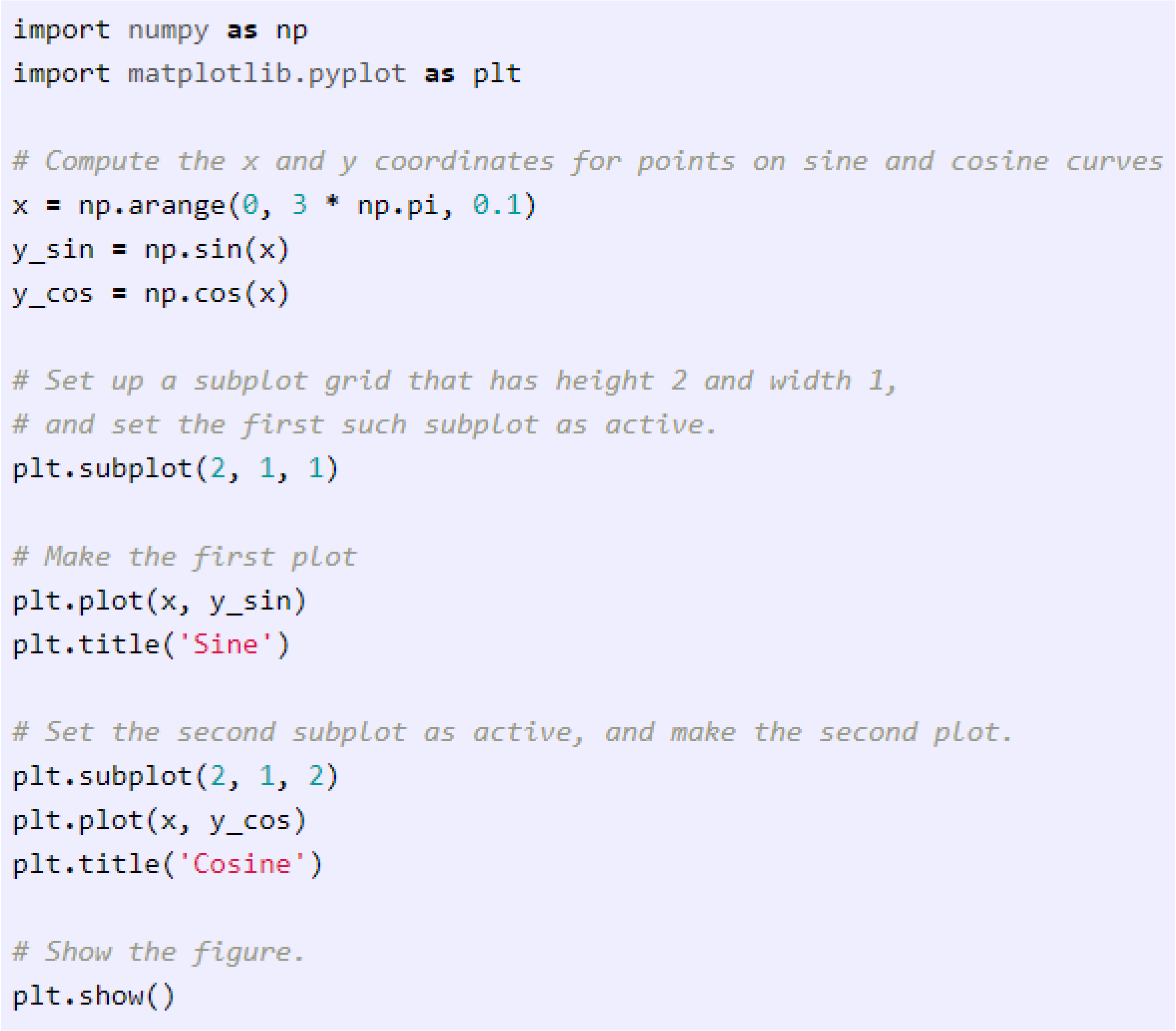


```
import numpy as np
import matplotlib.pyplot as plt

# 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()
```

python™

libraries Numpy SciPy Matplotlib

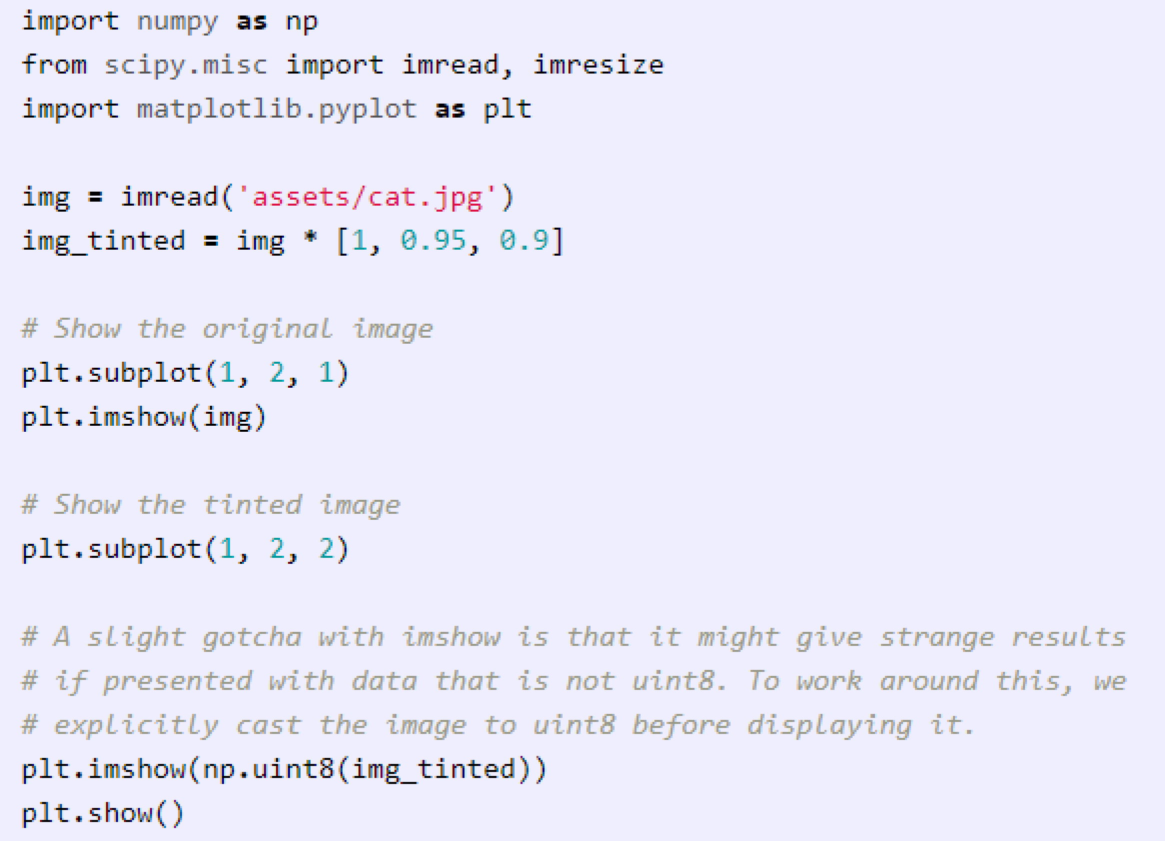
Matplotlib



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

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

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

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

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

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

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

Random processes

Vector processes

$$\mathbf{m}_x = E[\mathbf{x}(n)]$$

autocorrelation matrix $\mathbf{R_{xx}}(k)$

$$\mathbf{R_{xx}}(k) = E[\mathbf{x}(n)\mathbf{x}^{\dagger}(n-k)].$$

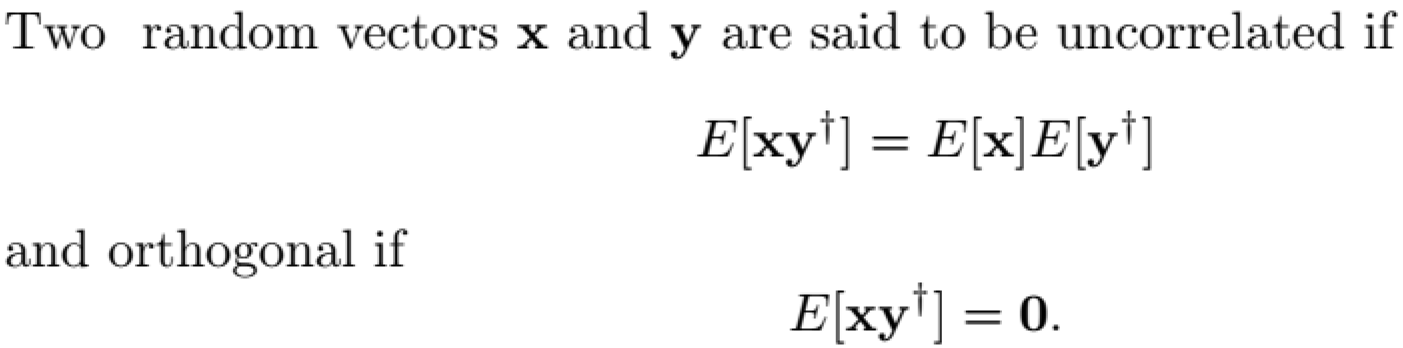
The Fourier transform of $\mathbf{R_{xx}}(k)$ is called the power spectrum or **PSD matrix**:

$$\mathbf{S_{xx}}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \mathbf{R_{xx}}(k)e^{-j\omega k}.$$

Random processes

Vector processes

Uncorrelatedness and orthogonality



Two random vectors $\mathbf{x}$ and $\mathbf{y}$ are said to be uncorrelated if

$$E[\mathbf{x}\mathbf{y}^\dagger] = E[\mathbf{x}]E[\mathbf{y}^\dagger]$$

and orthogonal if

$$E[\mathbf{x}\mathbf{y}^\dagger] = \mathbf{0}.$$

Random processes

Vector processes

Gaussian vector

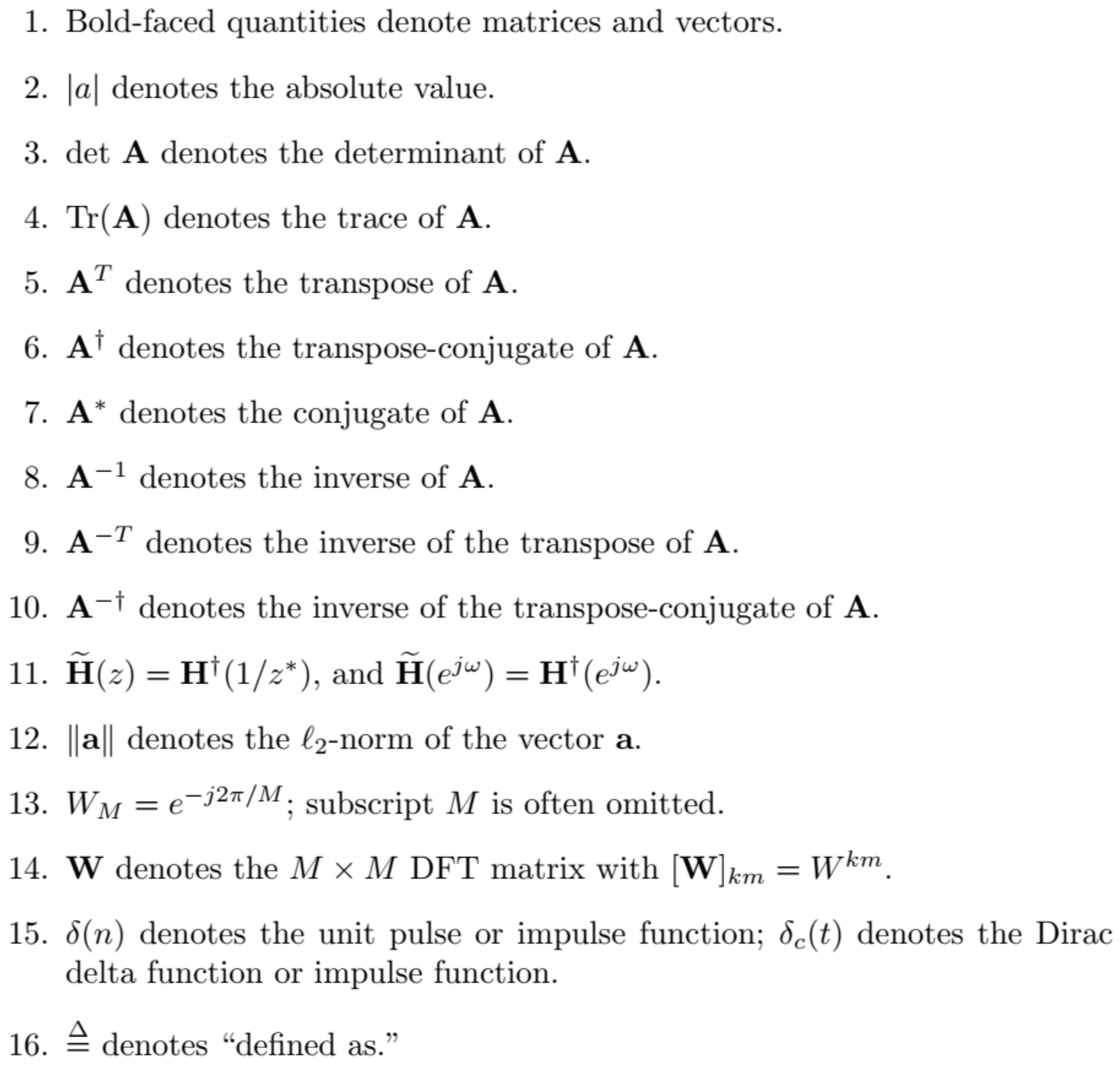
$$f(\mathbf{x}) = \frac{1}{\sqrt{\det(2\pi \mathbf{C}_{\mathbf{xx}})}} \, e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m}_{\mathbf{x}})^T \mathbf{C}_{\mathbf{xx}}^{-1} (\mathbf{x}-\mathbf{m}_{\mathbf{x}})}.$$

$$\mathbf{x} = [\, x_0 \quad x_1 \quad \ldots, x_{N-1} \,]^T \,,$$

$$f_{X_0}(x_0) = \int_{x_1} \ldots \int_{x_{N-1}} f(\mathbf{x}) dx_1 \ldots dx_{N-1}$$

Linear Algebra for Signal Processing

A Good Notation For Vectors & Matrices



1. Bold-faced quantities denote matrices and vectors.
2. $|a|$ denotes the absolute value.
3. det $\mathbf{A}$ denotes the determinant of $\mathbf{A}$.
4. $\mathrm{Tr}(\mathbf{A})$ denotes the trace of $\mathbf{A}$.
5. $\mathbf{A}^T$ denotes the transpose of $\mathbf{A}$.
6. $\mathbf{A}^\dagger$ denotes the transpose-conjugate of $\mathbf{A}$.
7. $\mathbf{A}^*$ denotes the conjugate of $\mathbf{A}$.
8. $\mathbf{A}^{-1}$ denotes the inverse of $\mathbf{A}$.
9. $\mathbf{A}^{-T}$ denotes the inverse of the transpose of $\mathbf{A}$.
10. $\mathbf{A}^{-\dagger}$ denotes the inverse of the transpose-conjugate of $\mathbf{A}$.
11. $\widetilde{\mathbf{H}}(z) = \mathbf{H}^\dagger(1/z^*)$, and $\widetilde{\mathbf{H}}(e^{j\omega}) = \mathbf{H}^\dagger(e^{j\omega})$.
12. $\|\mathbf{a}\|$ denotes the ℓ_2-norm of the vector $\mathbf{a}$.
13. $W_M = e^{-j2\pi/M}$; subscript M is often omitted.
14. $\mathbf{W}$ denotes the $M \times M$ DFT matrix with $[\mathbf{W}]_{km} = W^{km}$.
15. $\delta(n)$ denotes the unit pulse or impulse function; $\delta_c(t)$ denotes the Dirac delta function or impulse function.
16. $\triangleq$ denotes "defined as."

Inner products and norms

$$\mathbf{a} = [\,a_0 \quad a_1 \ldots a_{M-1}\,]^T, \quad \text{and} \quad \mathbf{b} = [\,b_0 \quad b_1 \ldots b_{M-1}\,]^T$$

$$< \mathbf{a}, \mathbf{b} > = \ \mathbf{b}^\dagger \mathbf{a} = \sum_{k=0}^{M-1} a_k b_k^*.$$

$$< f, g >= \int_0^T f(t) g^*(t) dt.$$

The vectors are said to be **orthogonal** if the inner product is zero, that is, $\mathbf{b}^\dagger \mathbf{a} = 0$.

$$\|\mathbf{a}\|_2 = (\mathbf{a}^\dagger \mathbf{a})^{1/2} = \left(\sum_{k=0}^{M-1} |a_k|^2 \right)^{1/2}$$

$$\|f(t)\|_2^2 = \int_0^T |f(t)|^2 dt.$$

$$\|\mathbf{a}\|_p = \left(\sum_{k=0}^{M-1} |a_k|^p \right)^{1/p}$$

$$\|f(t)\|_p = \left(\int_0^T |f(t)|^p dt \right)^{1/p}$$

Cauchy-Schwartz inequality

$$\Big|\sum_{k=0}^{M-1} a_k b_k^*\Big|^2 \leq \sum_{k=0}^{M-1} |a_k|^2 \sum_{k=0}^{M-1} |b_k|^2$$

$$\Big|\int_{-\infty}^{\infty} f(t)g^*(t)dt\Big|^2 \leq \int_{-\infty}^{\infty} |f(t)\ |^2 dt \int_{-\infty}^{\infty} |g(t)|^2 dt$$

$$|<\mathbf{a},\mathbf{b}>|^2 \leq \|\mathbf{a}\|_2^2 \|\mathbf{b}\|_2^2$$

$$|<f(t),g(t)>|^2 \leq \|f(t)\|_2^2 \|g(t)\|_2^2.$$

The AM-GM inequality

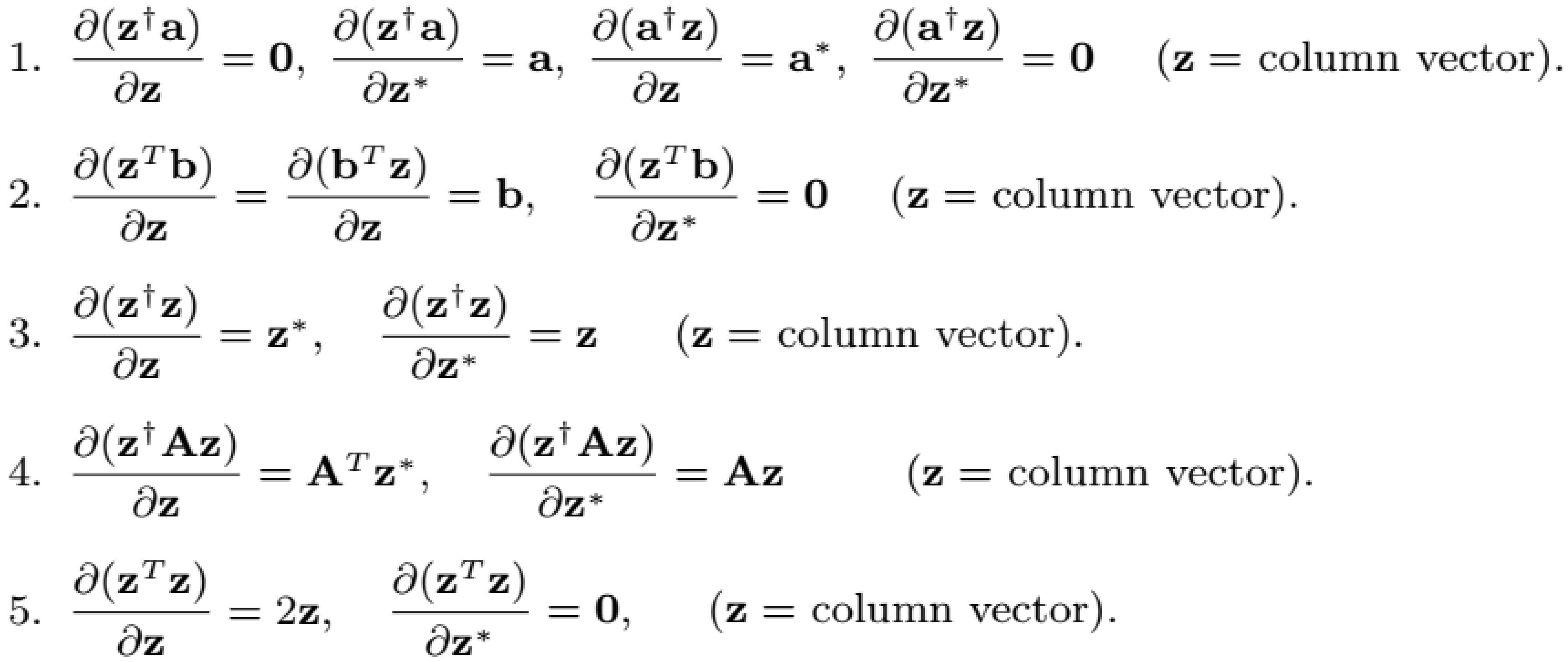
$$AM = \frac{1}{N}\sum_{k=0}^{N-1} a_k$$

$$GM = \Big(\prod_{k=0}^{N-1} a_k\Big)^{1/N}$$

For a set of positive numbers a_k, $0 \leq k \leq N-1$

$$AM \geq GM.$$

Derivatives in Optimization



1. $$\frac{\partial(\mathbf{z}^\dagger\mathbf{a})}{\partial\mathbf{z}} = \mathbf{0},\ \frac{\partial(\mathbf{z}^\dagger\mathbf{a})}{\partial\mathbf{z}^*} = \mathbf{a},\ \frac{\partial(\mathbf{a}^\dagger\mathbf{z})}{\partial\mathbf{z}} = \mathbf{a}^*,\ \frac{\partial(\mathbf{a}^\dagger\mathbf{z})}{\partial\mathbf{z}^*} = \mathbf{0} \quad (\mathbf{z} = \text{column vector}).$$

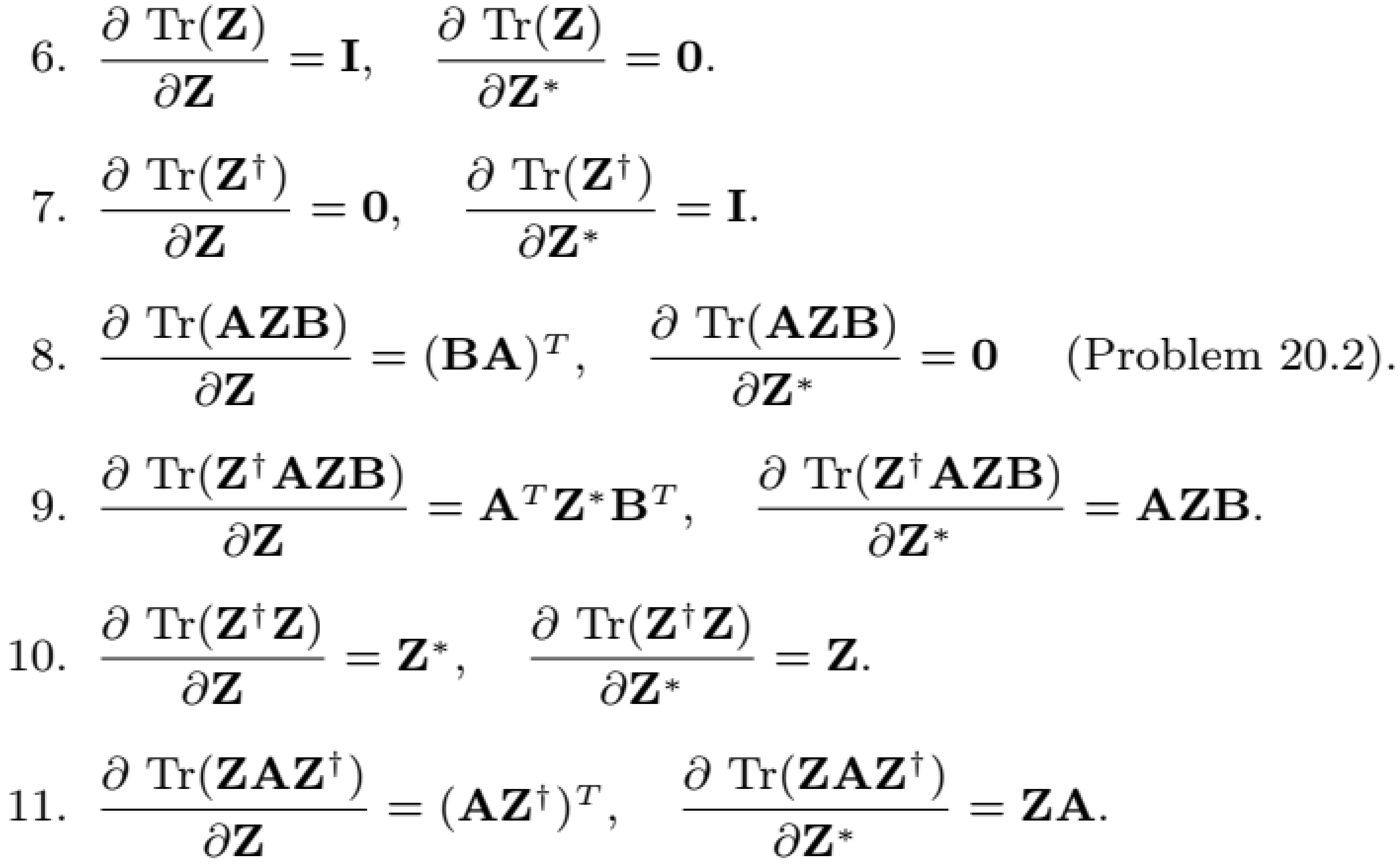
2. $$\frac{\partial(\mathbf{z}^T\mathbf{b})}{\partial\mathbf{z}} = \frac{\partial(\mathbf{b}^T\mathbf{z})}{\partial\mathbf{z}} = \mathbf{b}, \quad \frac{\partial(\mathbf{z}^T\mathbf{b})}{\partial\mathbf{z}^*} = \mathbf{0} \quad (\mathbf{z} = \text{column vector}).$$

3. $$\frac{\partial(\mathbf{z}^\dagger\mathbf{z})}{\partial\mathbf{z}} = \mathbf{z}^*, \quad \frac{\partial(\mathbf{z}^\dagger\mathbf{z})}{\partial\mathbf{z}^*} = \mathbf{z} \quad (\mathbf{z} = \text{column vector}).$$

4. $$\frac{\partial(\mathbf{z}^\dagger\mathbf{A}\mathbf{z})}{\partial\mathbf{z}} = \mathbf{A}^T\mathbf{z}^*, \quad \frac{\partial(\mathbf{z}^\dagger\mathbf{A}\mathbf{z})}{\partial\mathbf{z}^*} = \mathbf{A}\mathbf{z} \quad (\mathbf{z} = \text{column vector}).$$

5. $$\frac{\partial(\mathbf{z}^T\mathbf{z})}{\partial\mathbf{z}} = 2\mathbf{z}, \quad \frac{\partial(\mathbf{z}^T\mathbf{z})}{\partial\mathbf{z}^*} = \mathbf{0}, \quad (\mathbf{z} = \text{column vector}).$$

Derivatives in Optimization



6. $\dfrac{\partial \operatorname{Tr}(\mathbf{Z})}{\partial \mathbf{Z}} = \mathbf{I}, \quad \dfrac{\partial \operatorname{Tr}(\mathbf{Z})}{\partial \mathbf{Z}^*} = \mathbf{0}.$

7. $\dfrac{\partial \operatorname{Tr}(\mathbf{Z}^\dagger)}{\partial \mathbf{Z}} = \mathbf{0}, \quad \dfrac{\partial \operatorname{Tr}(\mathbf{Z}^\dagger)}{\partial \mathbf{Z}^*} = \mathbf{I}.$

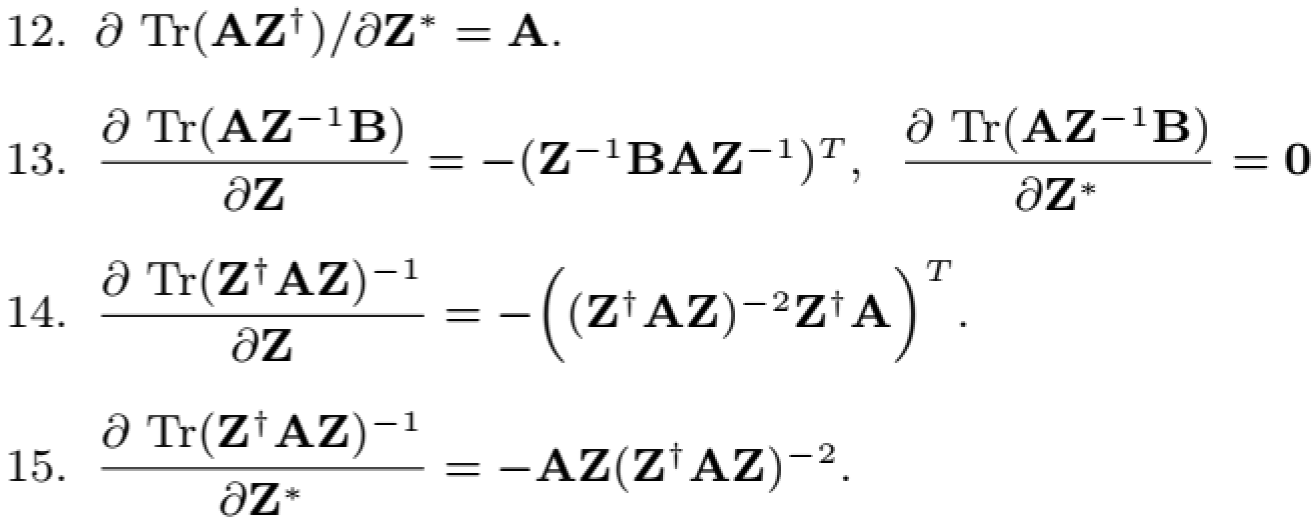
8. $\dfrac{\partial \operatorname{Tr}(\mathbf{AZB})}{\partial \mathbf{Z}} = (\mathbf{BA})^T, \quad \dfrac{\partial \operatorname{Tr}(\mathbf{AZB})}{\partial \mathbf{Z}^*} = \mathbf{0} \quad \text{(Problem 20.2).}$

9. $\dfrac{\partial \operatorname{Tr}(\mathbf{Z}^\dagger \mathbf{AZB})}{\partial \mathbf{Z}} = \mathbf{A}^T \mathbf{Z}^* \mathbf{B}^T, \quad \dfrac{\partial \operatorname{Tr}(\mathbf{Z}^\dagger \mathbf{AZB})}{\partial \mathbf{Z}^*} = \mathbf{AZB}.$

10. $\dfrac{\partial \operatorname{Tr}(\mathbf{Z}^\dagger \mathbf{Z})}{\partial \mathbf{Z}} = \mathbf{Z}^*, \quad \dfrac{\partial \operatorname{Tr}(\mathbf{Z}^\dagger \mathbf{Z})}{\partial \mathbf{Z}^*} = \mathbf{Z}.$

11. $\dfrac{\partial \operatorname{Tr}(\mathbf{ZAZ}^\dagger)}{\partial \mathbf{Z}} = (\mathbf{AZ}^\dagger)^T, \quad \dfrac{\partial \operatorname{Tr}(\mathbf{ZAZ}^\dagger)}{\partial \mathbf{Z}^*} = \mathbf{ZA}.$

Derivatives in Optimization



12. $\partial \, \mathrm{Tr}(\mathbf{A}\mathbf{Z}^{\dagger})/\partial \mathbf{Z}^{*} = \mathbf{A}.$

13. $\dfrac{\partial \, \mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}\mathbf{B})}{\partial \mathbf{Z}} = -(\mathbf{Z}^{-1}\mathbf{B}\mathbf{A}\mathbf{Z}^{-1})^{T}, \quad \dfrac{\partial \, \mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}\mathbf{B})}{\partial \mathbf{Z}^{*}} = \mathbf{0}$

14. $\dfrac{\partial \, \mathrm{Tr}(\mathbf{Z}^{\dagger}\mathbf{A}\mathbf{Z})^{-1}}{\partial \mathbf{Z}} = -\Big((\mathbf{Z}^{\dagger}\mathbf{A}\mathbf{Z})^{-2}\mathbf{Z}^{\dagger}\mathbf{A}\Big)^{T}.$

15. $\dfrac{\partial \, \mathrm{Tr}(\mathbf{Z}^{\dagger}\mathbf{A}\mathbf{Z})^{-1}}{\partial \mathbf{Z}^{*}} = -\mathbf{A}\mathbf{Z}(\mathbf{Z}^{\dagger}\mathbf{A}\mathbf{Z})^{-2}.$

Convexity, Schur convexity, and majorization theory

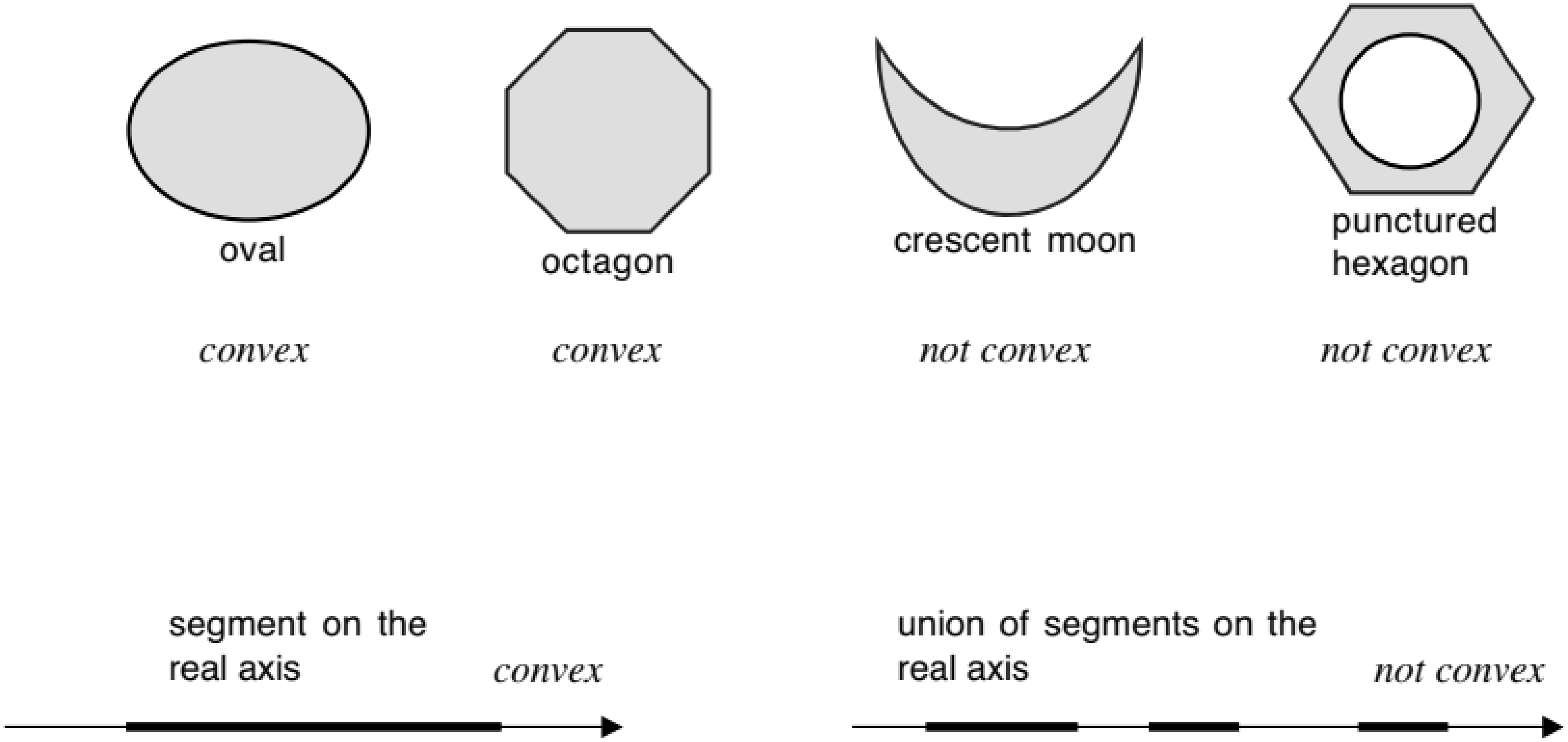


Figure 21.1. Examples of convex and non-convex sets.

Convexity, Schur convexity, and majorization theory

$$f\Big(\sum_{k=0}^{P-1}\alpha_k\mathbf{x}_k\Big) \le \sum_{k=0}^{P-1}\alpha_k f(\mathbf{x}_k)$$

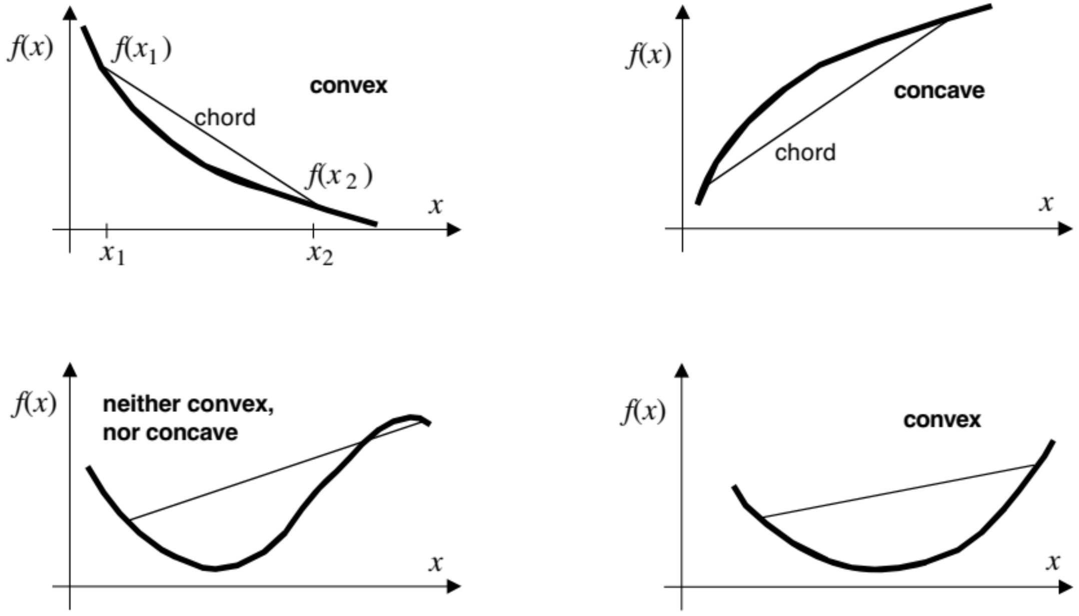
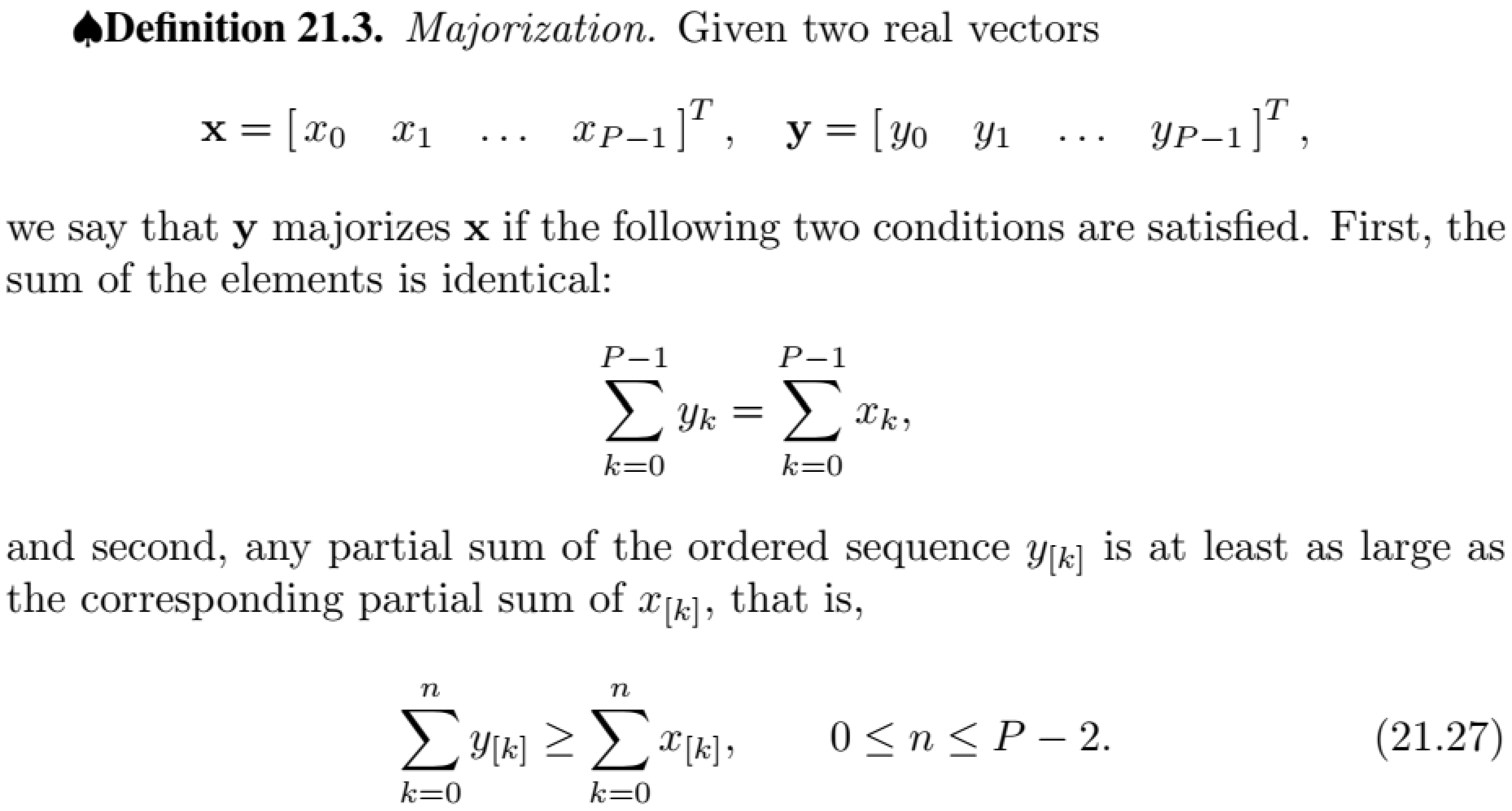


Figure 21.2. Examples of convex and concave functions.

Schur-convex functions



♠**Definition 21.3.** *Majorization.* Given two real vectors

$$\mathbf{x} = [\,x_0 \quad x_1 \quad \ldots \quad x_{P-1}\,]^T, \quad \mathbf{y} = [\,y_0 \quad y_1 \quad \ldots \quad y_{P-1}\,]^T,$$

we say that $\mathbf{y}$ majorizes $\mathbf{x}$ if the following two conditions are satisfied. First, the sum of the elements is identical:

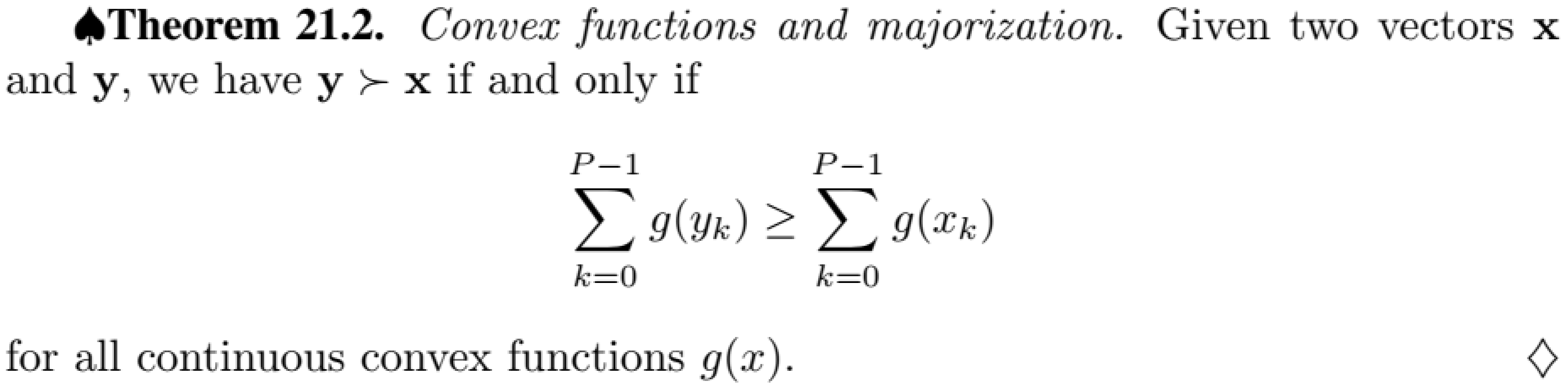
$$\sum_{k=0}^{P-1} y_k = \sum_{k=0}^{P-1} x_k,$$

and second, any partial sum of the ordered sequence $y_{[k]}$ is at least as large as the corresponding partial sum of $x_{[k]}$, that is,

$$\sum_{k=0}^{n} y_{[k]} \geq \sum_{k=0}^{n} x_{[k]}, \qquad 0 \leq n \leq P-2. \tag{21.27}$$

$$\mathbf{y} \succ \mathbf{x}$$

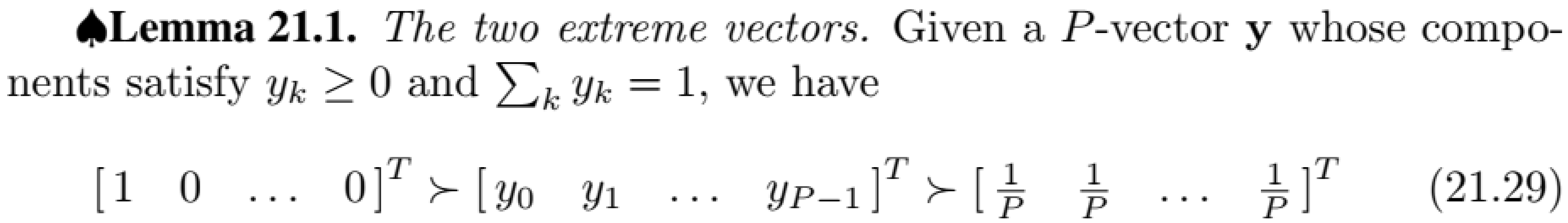
Schur-convex functions



♠**Theorem 21.2.** *Convex functions and majorization.* Given two vectors $\mathbf{x}$ and $\mathbf{y}$, we have $\mathbf{y} \succ \mathbf{x}$ if and only if

$$\sum_{k=0}^{P-1} g(y_k) \geq \sum_{k=0}^{P-1} g(x_k)$$

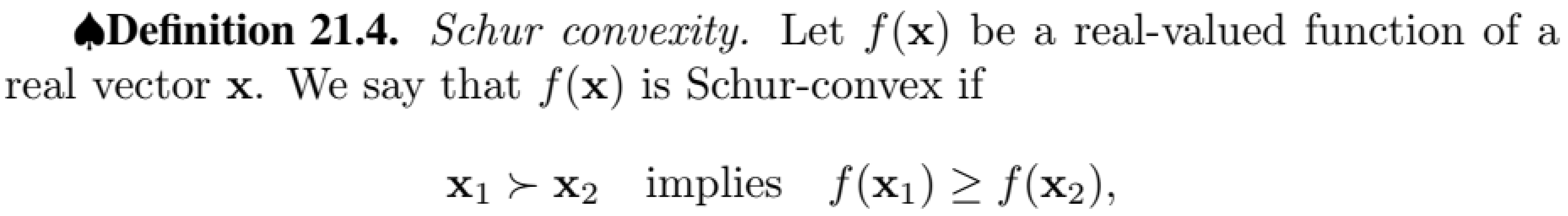
for all continuous convex functions $g(x)$. ◊



♠**Lemma 21.1.** *The two extreme vectors.* Given a P-vector $\mathbf{y}$ whose components satisfy $y_k \geq 0$ and $\sum_k y_k = 1$, we have

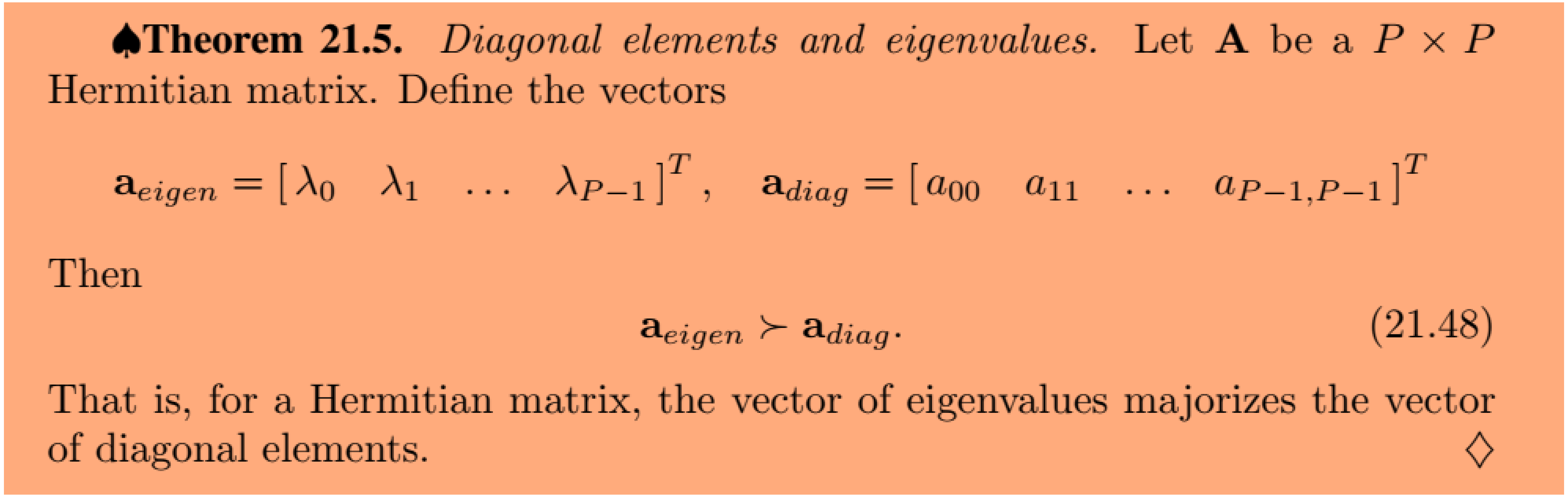
$$[1 \quad 0 \quad \ldots \quad 0]^T \succ [y_0 \quad y_1 \quad \ldots \quad y_{P-1}]^T \succ [\tfrac{1}{P} \quad \tfrac{1}{P} \quad \ldots \quad \tfrac{1}{P}]^T \qquad (21.29)$$

Schur-convex functions



♠**Definition 21.4.** *Schur convexity.* Let $f(\mathbf{x})$ be a real-valued function of a real vector $\mathbf{x}$. We say that $f(\mathbf{x})$ is Schur-convex if

$$\mathbf{x}_1 \succ \mathbf{x}_2 \quad \text{implies} \quad f(\mathbf{x}_1) \geq f(\mathbf{x}_2),$$



♠**Theorem 21.5.** *Diagonal elements and eigenvalues.* Let $\mathbf{A}$ be a $P \times P$ Hermitian matrix. Define the vectors

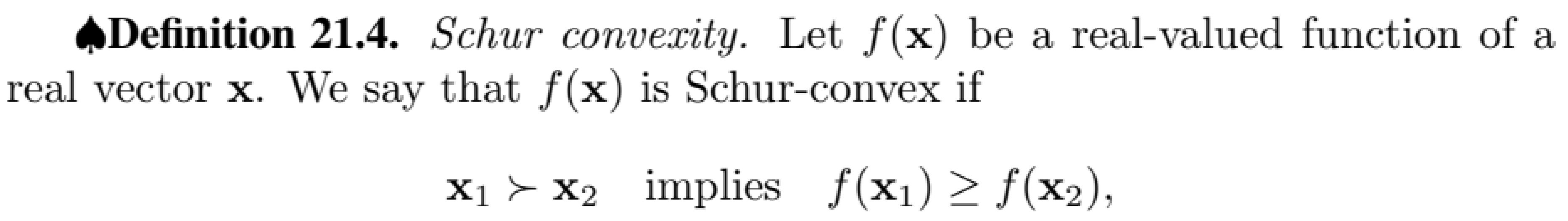
$$\mathbf{a}_{eigen} = [\lambda_0 \quad \lambda_1 \quad \ldots \quad \lambda_{P-1}]^T, \quad \mathbf{a}_{diag} = [a_{00} \quad a_{11} \quad \ldots \quad a_{P-1,P-1}]^T$$

Then

$$\mathbf{a}_{eigen} \succ \mathbf{a}_{diag}. \tag{21.48}$$

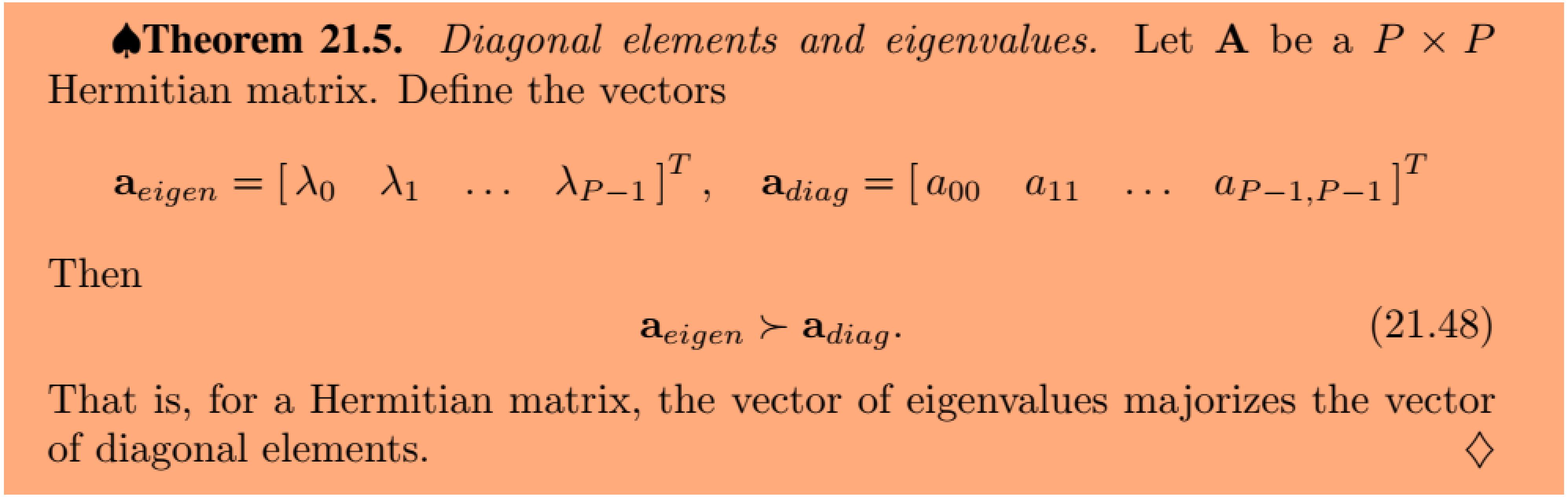
That is, for a Hermitian matrix, the vector of eigenvalues majorizes the vector of diagonal elements. ◊

Schur-convex functions



♠**Definition 21.4.** *Schur convexity.* Let $f(\mathbf{x})$ be a real-valued function of a real vector $\mathbf{x}$. We say that $f(\mathbf{x})$ is Schur-convex if

$$\mathbf{x}_1 \succ \mathbf{x}_2 \quad \text{implies} \quad f(\mathbf{x}_1) \geq f(\mathbf{x}_2),$$



♠**Theorem 21.5.** *Diagonal elements and eigenvalues.* Let $\mathbf{A}$ be a $P \times P$ Hermitian matrix. Define the vectors

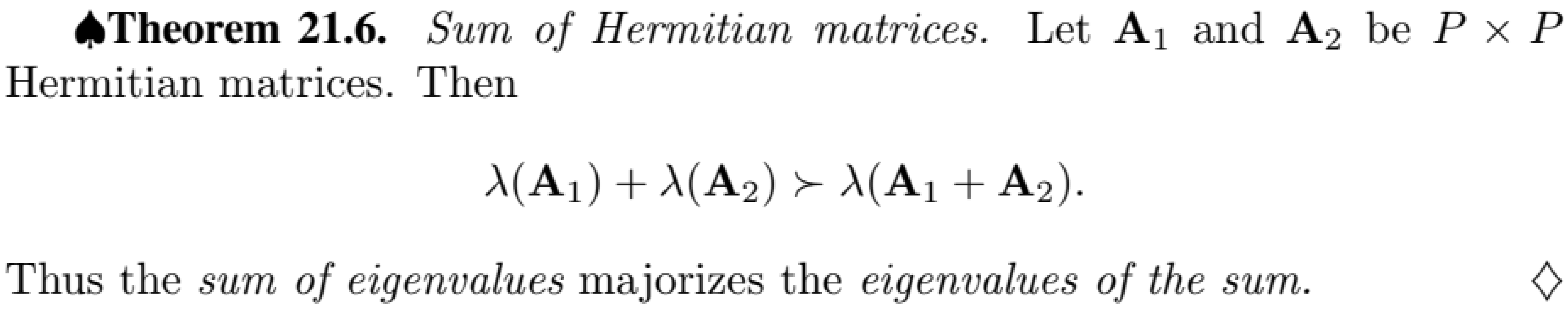
$$\mathbf{a}_{eigen} = [\,\lambda_0 \quad \lambda_1 \quad \ldots \quad \lambda_{P-1}\,]^T, \quad \mathbf{a}_{diag} = [\,a_{00} \quad a_{11} \quad \ldots \quad a_{P-1,P-1}\,]^T$$

Then

$$\mathbf{a}_{eigen} \succ \mathbf{a}_{diag}. \tag{21.48}$$

That is, for a Hermitian matrix, the vector of eigenvalues majorizes the vector of diagonal elements. ◊

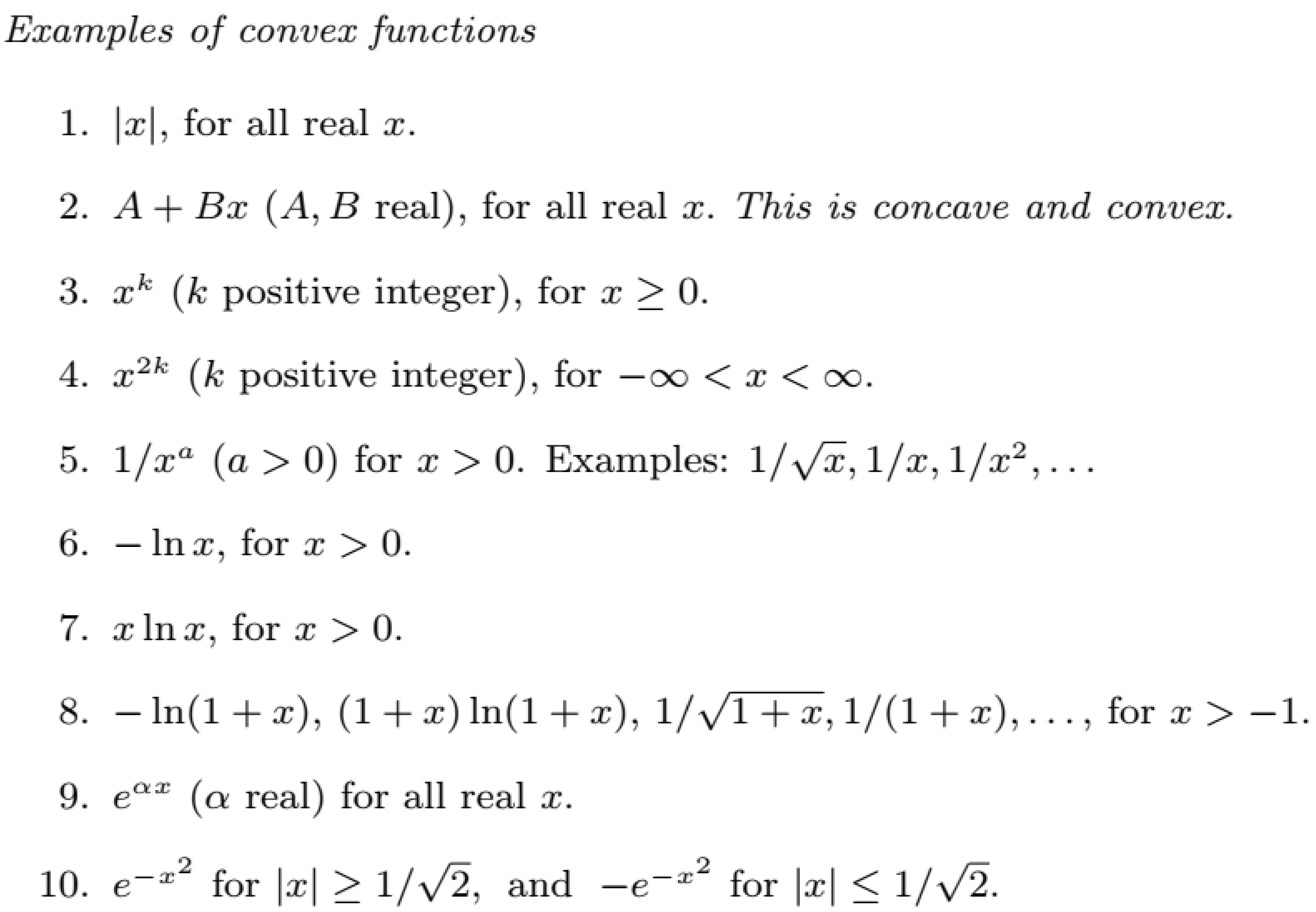
Schur-convex functions



♠**Theorem 21.6.** *Sum of Hermitian matrices.* Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be $P \times P$ Hermitian matrices. Then

$$\lambda(\mathbf{A}_1) + \lambda(\mathbf{A}_2) \succ \lambda(\mathbf{A}_1 + \mathbf{A}_2).$$

Thus the *sum of eigenvalues* majorizes the *eigenvalues of the sum.* ◊



Examples of convex functions

1. $|x|$, for all real x.
2. $A + Bx$ (A, B real), for all real x. *This is concave and convex.*
3. x^k (k positive integer), for $x \geq 0$.
4. x^{2k} (k positive integer), for $-\infty < x < \infty$.
5. $1/x^a$ $(a > 0)$ for $x > 0$. Examples: $1/\sqrt{x}, 1/x, 1/x^2, \ldots$
6. $-\ln x$, for $x > 0$.
7. $x \ln x$, for $x > 0$.
8. $-\ln(1+x)$, $(1+x)\ln(1+x)$, $1/\sqrt{1+x}, 1/(1+x), \ldots,$ for $x > -1$.
9. $e^{\alpha x}$ (α real) for all real x.
10. e^{-x^2} for $|x| \geq 1/\sqrt{2}$, and $-e^{-x^2}$ for $|x| \leq 1/\sqrt{2}$.

11. $\mathrm{erfc}(K/\sqrt{x})$ in $0 < x \leq 2K^2/3$ and $-\mathrm{erfc}(K/\sqrt{x})$ in $x \geq 2K^2/3$.

12. $\mathcal{Q}(A/\sqrt{x})$ in $0 < x \leq A^2/3$ and $-\mathcal{Q}(A/\sqrt{x})$ in $x \geq A^2/3$.

13. $A + Bx_0 + Cx_1$ (A, B, C real), for real x_0, x_1. *This is concave and convex.*

14. $-\ln(x_0 x_1)$, for $x_k > 0$. *Note:* The product $x_0 x_1$ is neither convex nor concave.

15. $-\prod_{k=0}^{P-1}(x_k)^{1/P}$ for $x_k > 0$ (geometric mean is *concave*; Sec. 21.2.4).

16. ℓ_p norm $\|\mathbf{x}\|_p \overset{\Delta}{=} (\sum_{k=0}^{P-1} |x_k|^p)^{1/p}$, $p > 1$ for all real $\mathbf{x}$.

Properties of convex functions

1. $f(\mathbf{x})$ is convex $\Leftrightarrow -f(\mathbf{x})$ is concave.

2. $f(x)$ convex $\Leftrightarrow d^2f(x)/dx^2 \geq 0$ (assuming $f(x)$ is twice differentiable).

3. $f(\mathbf{x})$ convex $\Leftrightarrow$ Hessian $\geq \mathbf{0}$ (assuming Hessian exists; Sec. 21.2.1).

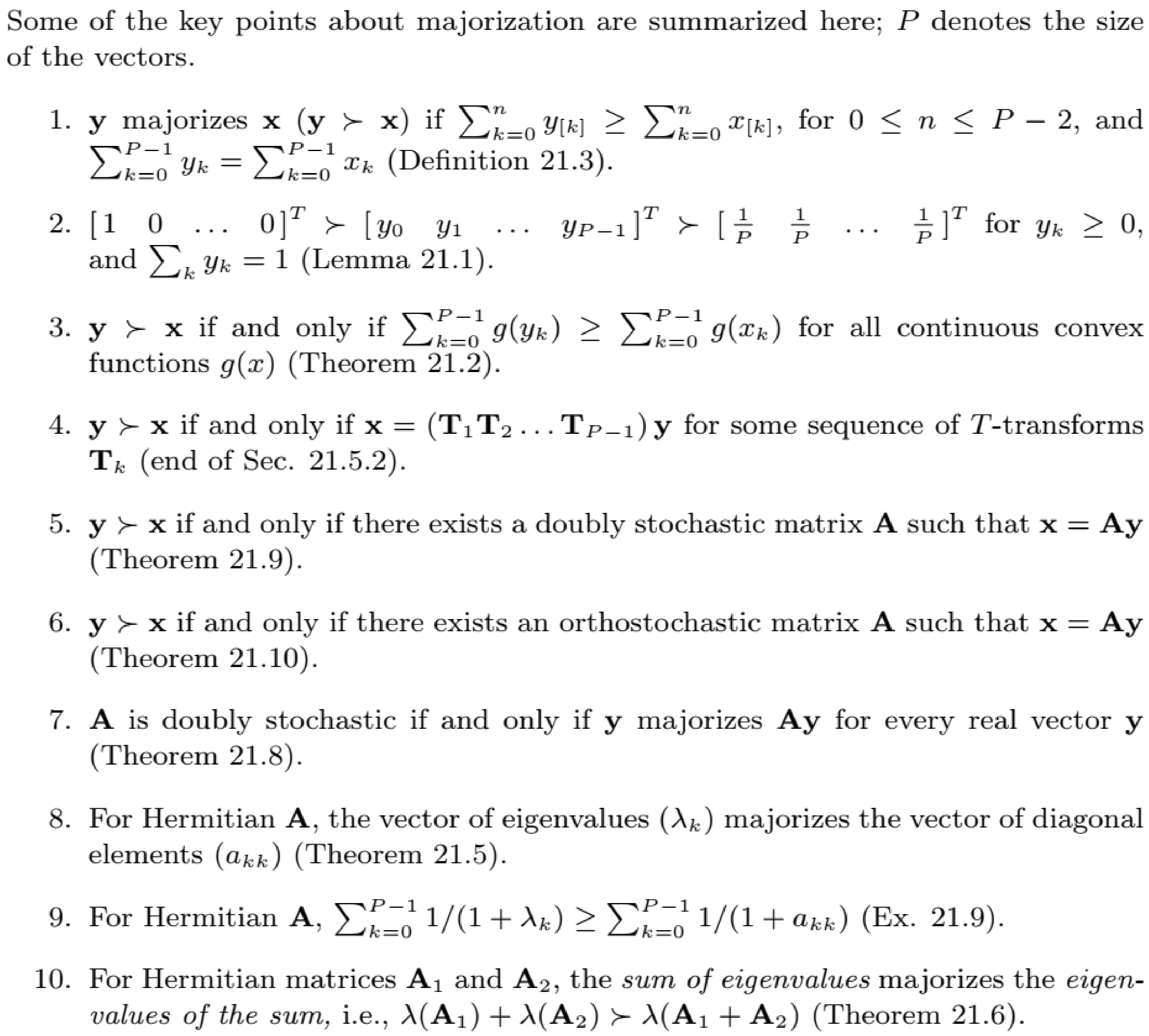
4. $f(\mathbf{x})$ convex $\Leftrightarrow f(\sum_{k=0}^{P-1} \alpha_k \mathbf{x}_k) \leq \sum_{k=0}^{P-1} \alpha_k f(\mathbf{x}_k)$ (see Eq. (21.2)).

5. $f(x)$ convex $\Rightarrow f(E[X]) \leq E[f(X)]$ (Jensen's inequality, Eq. (21.25)).

6. $f_k(\mathbf{x})$ convex on $\mathcal{A} \Rightarrow f(\mathbf{x}) \stackrel{\Delta}{=} \max_k f_k(\mathbf{x})$ is convex on $\mathcal{A}$ (Sec. 21.2.7).

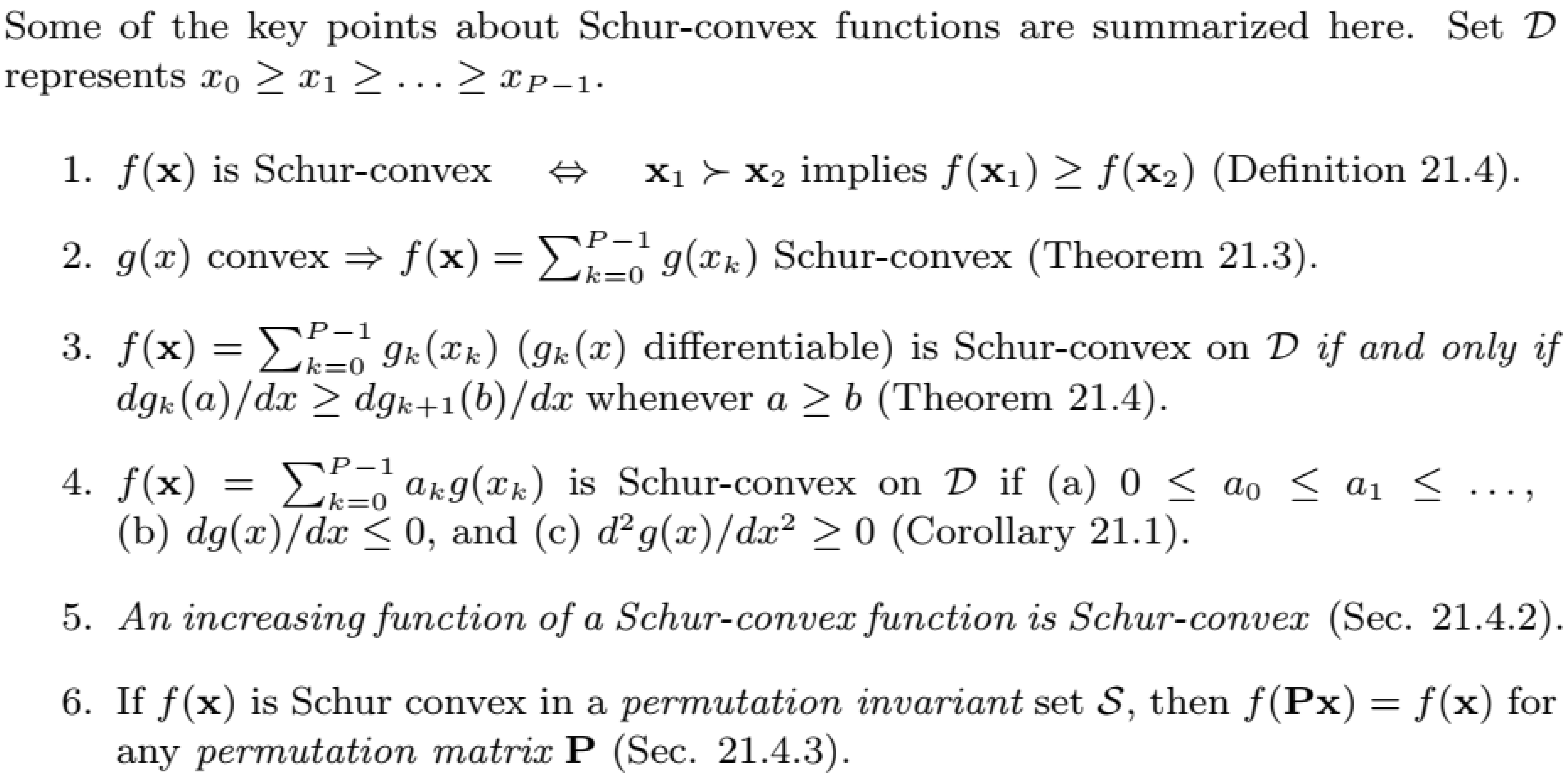
7. $f_k(x)$ convex on $a_k \leq x \leq b_k \Rightarrow g(\mathbf{x}) = \sum_{k=0}^{P-1} f_k(x_k)$ convex on $a_k \leq x_k \leq b_k$ (Sec. 21.2.7).

8. Increasing convex functions of convex functions are convex (Theorem 21.1).



Some of the key points about majorization are summarized here; P denotes the size of the vectors.

1. $\mathbf{y}$ majorizes $\mathbf{x}$ $(\mathbf{y} \succ \mathbf{x})$ if $\sum_{k=0}^{n} y_{[k]} \geq \sum_{k=0}^{n} x_{[k]}$, for $0 \leq n \leq P-2$, and $\sum_{k=0}^{P-1} y_k = \sum_{k=0}^{P-1} x_k$ (Definition 21.3).
2. $[1 \quad 0 \quad \ldots \quad 0]^T \succ [y_0 \quad y_1 \quad \ldots \quad y_{P-1}]^T \succ [\frac{1}{P} \quad \frac{1}{P} \quad \ldots \quad \frac{1}{P}]^T$ for $y_k \geq 0$, and $\sum_k y_k = 1$ (Lemma 21.1).
3. $\mathbf{y} \succ \mathbf{x}$ if and only if $\sum_{k=0}^{P-1} g(y_k) \geq \sum_{k=0}^{P-1} g(x_k)$ for all continuous convex functions $g(x)$ (Theorem 21.2).
4. $\mathbf{y} \succ \mathbf{x}$ if and only if $\mathbf{x} = (\mathbf{T}_1\mathbf{T}_2 \ldots \mathbf{T}_{P-1})\,\mathbf{y}$ for some sequence of T-transforms $\mathbf{T}_k$ (end of Sec. 21.5.2).
5. $\mathbf{y} \succ \mathbf{x}$ if and only if there exists a doubly stochastic matrix $\mathbf{A}$ such that $\mathbf{x} = \mathbf{A}\mathbf{y}$ (Theorem 21.9).
6. $\mathbf{y} \succ \mathbf{x}$ if and only if there exists an orthostochastic matrix $\mathbf{A}$ such that $\mathbf{x} = \mathbf{A}\mathbf{y}$ (Theorem 21.10).
7. $\mathbf{A}$ is doubly stochastic if and only if $\mathbf{y}$ majorizes $\mathbf{A}\mathbf{y}$ for every real vector $\mathbf{y}$ (Theorem 21.8).
8. For Hermitian $\mathbf{A}$, the vector of eigenvalues (λ_k) majorizes the vector of diagonal elements (a_{kk}) (Theorem 21.5).
9. For Hermitian $\mathbf{A}$, $\sum_{k=0}^{P-1} 1/(1+\lambda_k) \geq \sum_{k=0}^{P-1} 1/(1+a_{kk})$ (Ex. 21.9).
10. For Hermitian matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, the *sum of eigenvalues* majorizes the *eigenvalues of the sum*, i.e., $\lambda(\mathbf{A}_1) + \lambda(\mathbf{A}_2) \succ \lambda(\mathbf{A}_1 + \mathbf{A}_2)$ (Theorem 21.6).



Some of the key points about Schur-convex functions are summarized here. Set $\mathcal{D}$ represents $x_0 \geq x_1 \geq \ldots \geq x_{P-1}$.

1. $f(\mathbf{x})$ is Schur-convex $\quad \Leftrightarrow \quad \mathbf{x}_1 \succ \mathbf{x}_2$ implies $f(\mathbf{x}_1) \geq f(\mathbf{x}_2)$ (Definition 21.4).

2. $g(x)$ convex $\Rightarrow f(\mathbf{x}) = \sum_{k=0}^{P-1} g(x_k)$ Schur-convex (Theorem 21.3).

3. $f(\mathbf{x}) = \sum_{k=0}^{P-1} g_k(x_k)$ ($g_k(x)$ differentiable) is Schur-convex on $\mathcal{D}$ *if and only if* $dg_k(a)/dx \geq dg_{k+1}(b)/dx$ whenever $a \geq b$ (Theorem 21.4).

4. $f(\mathbf{x}) = \sum_{k=0}^{P-1} a_k g(x_k)$ is Schur-convex on $\mathcal{D}$ if (a) $0 \leq a_0 \leq a_1 \leq \ldots$, (b) $dg(x)/dx \leq 0$, and (c) $d^2g(x)/dx^2 \geq 0$ (Corollary 21.1).

5. *An increasing function of a Schur-convex function is Schur-convex* (Sec. 21.4.2).

6. If $f(\mathbf{x})$ is Schur convex in a *permutation invariant* set $\mathcal{S}$, then $f(\mathbf{Px}) = f(\mathbf{x})$ for any *permutation matrix* $\mathbf{P}$ (Sec. 21.4.3).

Set $\mathcal{D}$ is $x_0 \geq x_1 \geq \ldots \geq x_{P-1}$, set $\mathcal{D}_+$ is $x_0 \geq x_1 \geq \ldots \geq x_{P-1} \geq 0$, and set $\mathcal{D}_{++}$ is $x_0 \geq x_1 \geq \ldots \geq x_{P-1} > 0$. It is assumed throughout that $0 \leq a_0 \leq a_1 \leq \ldots \leq a_{P-1}$. Here are some examples of Schur-convex functions.

1. $\exp(\sum_{k=0}^{P-1} 1/x_k)$ for $x_k > 0$.
2. $\sum_{k=0}^{P-1} e^{-x_k^2}$ in $|x_k| \geq 1/\sqrt{2}$ and $-\sum_{k=0}^{P-1} e^{-x_k^2}$ in $|x_k| \leq 1/\sqrt{2}$.
3. $\sum_{k=0}^{P-1} \operatorname{erfc}(1/\sqrt{x_k})$ in $0 < x_k \leq 2/3$.
4. $\max_k\{x_k\}$ for any real $\mathbf{x}$.
5. $\sum_{k=0}^{P-1} a_k/x_k^p$ $(p > 0)$ in $\mathcal{D}_{++}$
6. Examples of above: $\sum_{k=0}^{P-1} a_k/\sqrt{x_k}$, $\sum_{k=0}^{P-1} a_k/x_k$, $\sum_{k=0}^{P-1} a_k/x_k^2$ in $\mathcal{D}_+$
7. $\sum_{k=0}^{P-1} a_k/(1+x_k)$ in $\mathcal{D}_+$
8. $\sum_{k=0}^{P-1} a_k e^{-\alpha x_k}$ $(\alpha > 0)$ in $\mathcal{D}_+$
9. $-\sum_{k=0}^{P-1} a_k \ln x_k$ in $\mathcal{D}_{++}$
10. $-\prod_{k=0}^{P-1} x_k^{a_k}$ in $\mathcal{D}_{++}$
11. $\sum_{k=0}^{P-1} a_k/(1+x_k)^2$ is Schur convex in $x_0 \geq x_1 \geq \ldots \geq x_{P-1} \geq 1/\sqrt{3}$.
12. $-\sum_{k=0}^{P-1} a_k x_k$ in $\mathcal{D}$.
13. $\sum_{k=0}^{P-1} a_k(1-x_k)/x_k$ in $\mathcal{D}_{++}$
14. $\sum_{k=0}^{P-1} a_k \ln[(1-x_k)/x_k]$ in $\mathcal{D}_{++}$ if $x_k \leq 0.5$.
15. $\prod_{k=0}^{P-1}[(1-x_k)/x_k]^{a_k}$ in $\mathcal{D}_{++}$ if $x_k \leq 0.5$.

Optimization with equality and inequality constraints

$$\text{minimize} \quad f(\mathbf{x}) \triangleq f(x_0, x_1, \ldots, x_{N-1}),$$

$$h_k(\mathbf{x}) = 0, \quad 0 \le k \le M-1 \quad \text{and} \quad g_k(\mathbf{x}) \le 0, \quad 0 \le k \le P-1$$

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_0(\mathbf{x}) \\ h_1(\mathbf{x}) \\ \vdots \\ h_{M-1}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_0(\mathbf{x}) \\ g_1(\mathbf{x}) \\ \vdots \\ g_{P-1}(\mathbf{x}) \end{bmatrix}.$$

$$\bigtriangledown f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_0} & \frac{\partial f(\mathbf{x})}{\partial x_1} & \ldots & \frac{\partial f(\mathbf{x})}{\partial x_{N-1}} \end{bmatrix}$$

$$\bigtriangledown \mathbf{h}(\mathbf{x}) = \begin{bmatrix} \frac{\partial h_0(\mathbf{x})}{\partial x_0} & \frac{\partial h_0(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_0(\mathbf{x})}{\partial x_{N-1}} \\ \frac{\partial h_1(\mathbf{x})}{\partial x_0} & \frac{\partial h_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_1(\mathbf{x})}{\partial x_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_{M-1}(\mathbf{x})}{\partial x_0} & \frac{\partial h_{M-1}(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_{M-1}(\mathbf{x})}{\partial x_{N-1}} \end{bmatrix}$$

$$\bigtriangledown \mathbf{g}(\mathbf{x}) = \begin{bmatrix} \frac{\partial g_0(\mathbf{x})}{\partial x_0} & \frac{\partial g_0(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial g_0(\mathbf{x})}{\partial x_{N-1}} \\ \frac{\partial g_1(\mathbf{x})}{\partial x_0} & \frac{\partial g_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial g_1(\mathbf{x})}{\partial x_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{P-1}(\mathbf{x})}{\partial x_0} & \frac{\partial g_{P-1}(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial g_{P-1}(\mathbf{x})}{\partial x_{N-1}} \end{bmatrix}$$

Lagrange multipliers λ_k

$$\mathcal{L} = \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{M-1} \end{bmatrix}$$

KKT multipliers

$$\mathcal{M} = \begin{bmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_{P-1} \end{bmatrix}$$

Optimization with equality and inequality constraints

Karush–Kuhn–Tucker (KKT) conditions

1. ***Non-negativity***. $\mathcal{M} \geq \mathbf{0}$;
2. ***Stationarity***. $\bigtriangledown f(\widehat{\mathbf{x}}) + \mathcal{L}^T \bigtriangledown \mathbf{h}(\widehat{\mathbf{x}}) + \mathcal{M}^T \bigtriangledown \mathbf{g}(\widehat{\mathbf{x}}) = \mathbf{0}$;
3. ***Orthogonality***. $\mathcal{M}^T \mathbf{g}(\widehat{\mathbf{x}}) = 0$.

$$\mu_\ell \geq 0, \quad 0 \leq \ell \leq P-1$$

$$\frac{\partial f(\widehat{\mathbf{x}})}{\partial x_k} + \sum_{\ell=0}^{M-1} \lambda_\ell \frac{\partial h_\ell(\widehat{\mathbf{x}})}{\partial x_k} + \sum_{\ell=0}^{P-1} \mu_\ell \frac{\partial g_\ell(\widehat{\mathbf{x}})}{\partial x_k} = 0, \quad 0 \leq k \leq N-1$$

$$\sum_{\ell=0}^{P-1} \mu_\ell g_\ell(\widehat{\mathbf{x}}) = 0$$

Optimization with equality and inequality constraints

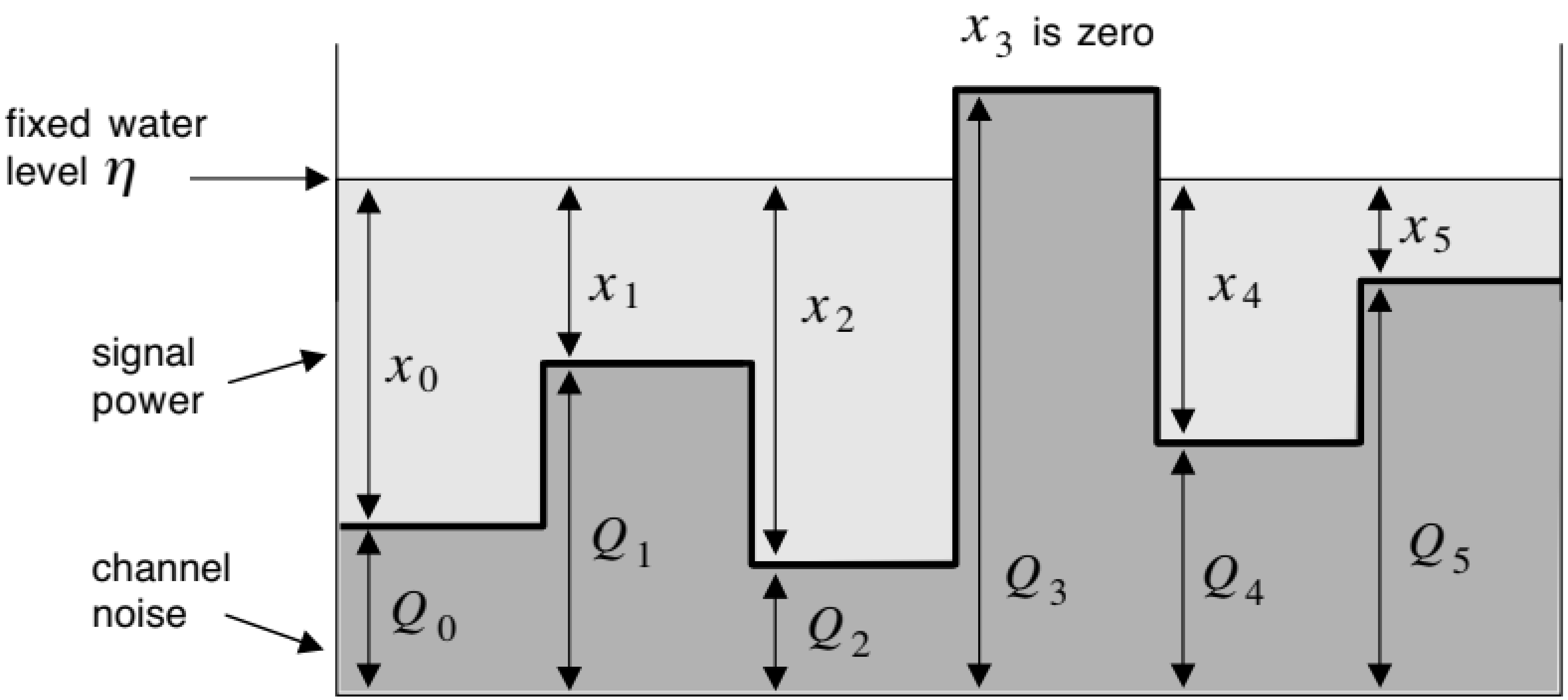


Figure 22.3. Optimizing capacity by power allocation. This is called the water-pouring solution.

Matrices

$$\mathbf{P} = \begin{bmatrix} 1 & 2+j \\ 1 & 3 \\ 2 & 4-j \end{bmatrix}$$

$$\mathbf{P}^T = \begin{bmatrix} 1 & 1 & 2 \\ 2+j & 3 & 4-j \end{bmatrix}, \quad \mathbf{P}^* = \begin{bmatrix} 1 & 2-j \\ 1 & 3 \\ 2 & 4+j \end{bmatrix}, \quad \mathbf{P}^\dagger = \begin{bmatrix} 1 & 1 & 2 \\ 2-j & 3 & 4+j \end{bmatrix}.$$

An $M \times M$ diagonal matrix with all diagonal elements equal to unity is called the **identity** matrix, and is indicated as $\mathbf{I}_M$ or simply $\mathbf{I}$.

upper triangular matrix:

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 5 \end{bmatrix}$$

Matrices

$\mathbf{P}$ is $N \times M$ and $\mathbf{Q}$ is $M \times K$

$$[\mathbf{PQ}]_{nk} = \sum_{m=0}^{M-1} p_{nm} q_{mk}.$$

$$\mathbf{PQ} \neq \mathbf{QP}$$

Determinant and trace

$$\operatorname{Tr}(\mathbf{P}) = \sum_{k} p_{kk}.$$

$$\operatorname{Tr}(\mathbf{PQ}) = \operatorname{Tr}(\mathbf{QP})$$

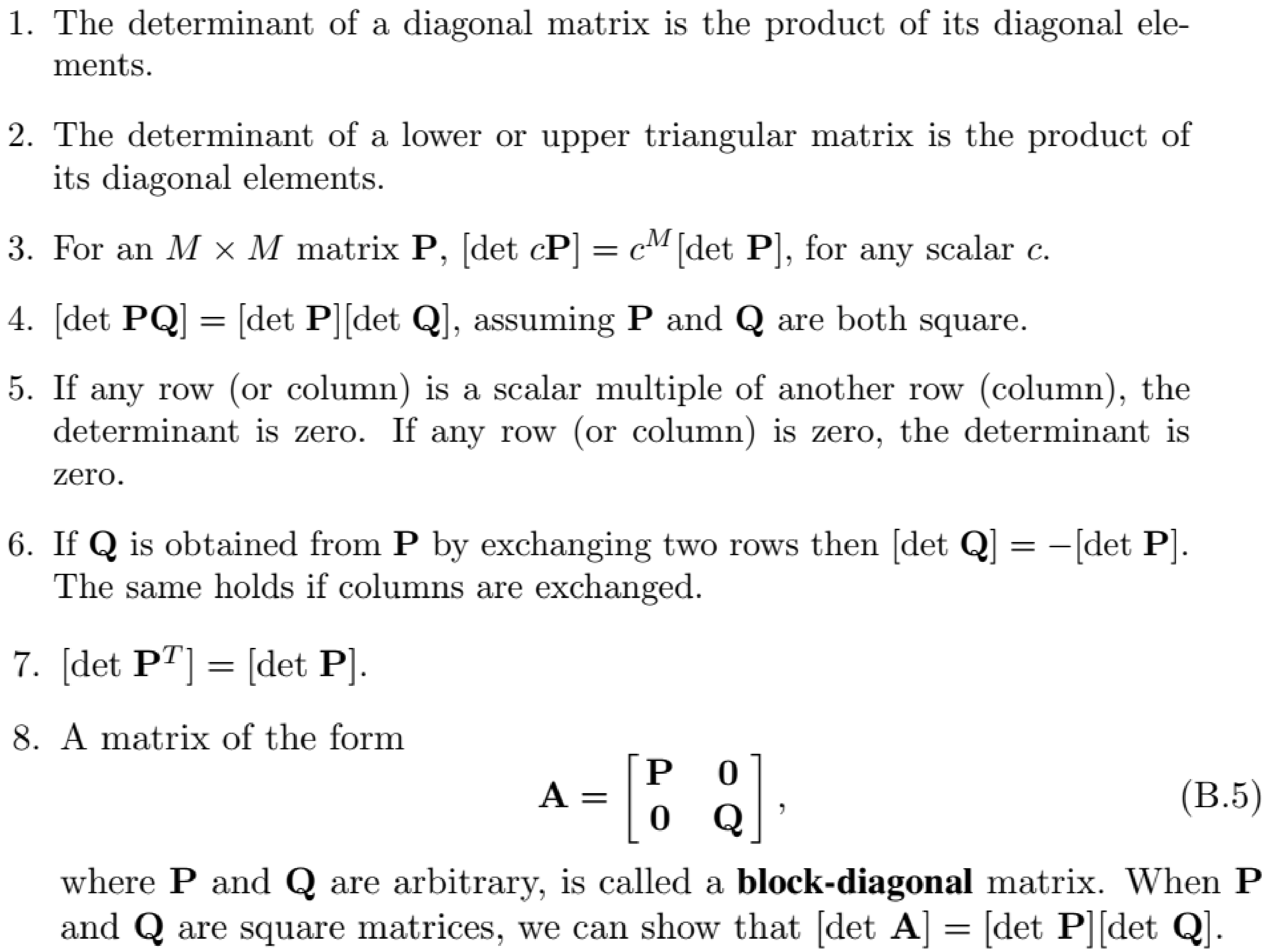
$$\det \mathbf{P} = \sum_{k=0}^{M-1} (-1)^{k+m} p_{km} M_{km},$$

$(-1)^{k+m} M_{km}$ is said to be the **cofactor** of p_{km}.

singular if $[\det \mathbf{P}] = 0$, and **nonsingular** if $[\det \mathbf{P}] \neq 0$.

Matrices

Properties of determinants



1. The determinant of a diagonal matrix is the product of its diagonal elements.
2. The determinant of a lower or upper triangular matrix is the product of its diagonal elements.
3. For an $M \times M$ matrix $\mathbf{P}$, $[\det c\mathbf{P}] = c^M [\det \mathbf{P}]$, for any scalar c.
4. $[\det \mathbf{PQ}] = [\det \mathbf{P}][\det \mathbf{Q}]$, assuming $\mathbf{P}$ and $\mathbf{Q}$ are both square.
5. If any row (or column) is a scalar multiple of another row (column), the determinant is zero. If any row (or column) is zero, the determinant is zero.
6. If $\mathbf{Q}$ is obtained from $\mathbf{P}$ by exchanging two rows then $[\det \mathbf{Q}] = -[\det \mathbf{P}]$. The same holds if columns are exchanged.
7. $[\det \mathbf{P}^T] = [\det \mathbf{P}]$.
8. A matrix of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}, \tag{B.5}$$

   where $\mathbf{P}$ and $\mathbf{Q}$ are arbitrary, is called a **block-diagonal** matrix. When $\mathbf{P}$ and $\mathbf{Q}$ are square matrices, we can show that $[\det \mathbf{A}] = [\det \mathbf{P}][\det \mathbf{Q}]$.

Matrices

Rank

The rank of a matrix $\mathbf{P}$ is equal to the number of linearly independent rows in the matrix.

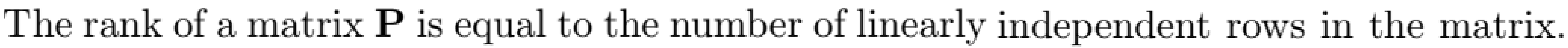
1. An $M \times M$ matrix is nonsingular (i.e., its determinant is nonzero) if and only if it has full rank, that is, its rank is M.
2. Let $\mathbf{A}$ and $\mathbf{B}$ be $M \times N$ and $N \times K$ matrices with ranks ρ_a and ρ_b. Let ρ be the rank of $\mathbf{AB}$. Then,

   $$\rho_a + \rho_b - N \leq \rho \leq \min(\rho_a, \rho_b).$$

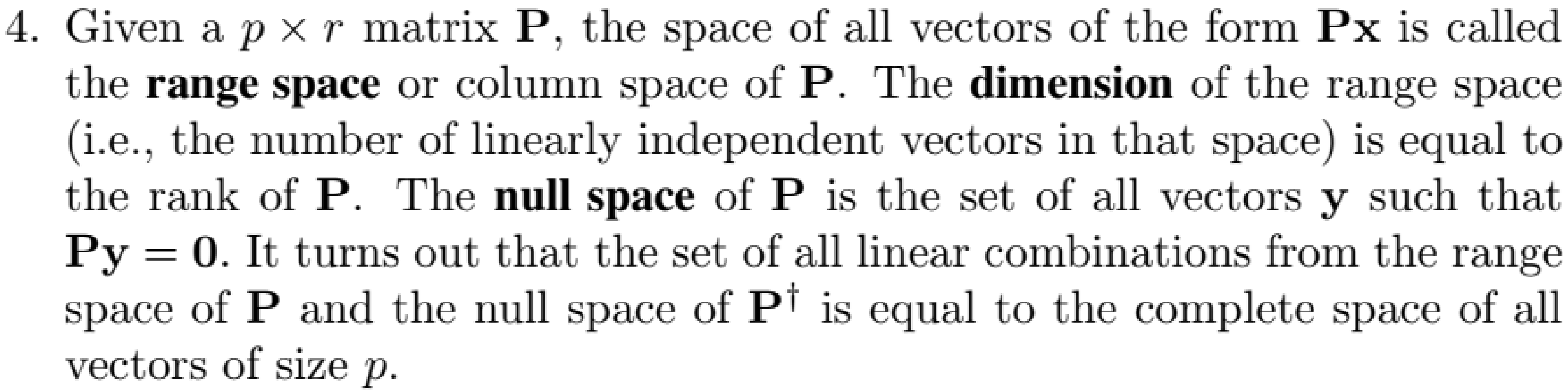
   This is called **Sylvester's inequality.**
3. Given two square matrices $\mathbf{A}$ and $\mathbf{B}$, the matrices $\mathbf{I} - \mathbf{AB}$ and $\mathbf{I} - \mathbf{BA}$ have identical rank. However, the products $\mathbf{AB}$ and $\mathbf{BA}$ may not have the same rank in general.

Matrices

Rank



The rank of a matrix $\mathbf{P}$ is equal to the number of linearly independent rows in the matrix.



4. Given a $p \times r$ matrix $\mathbf{P}$, the space of all vectors of the form $\mathbf{Px}$ is called the **range space** or column space of $\mathbf{P}$. The **dimension** of the range space (i.e., the number of linearly independent vectors in that space) is equal to the rank of $\mathbf{P}$. The **null space** of $\mathbf{P}$ is the set of all vectors $\mathbf{y}$ such that $\mathbf{Py} = \mathbf{0}$. It turns out that the set of all linear combinations from the range space of $\mathbf{P}$ and the null space of $\mathbf{P}^\dagger$ is equal to the complete space of all vectors of size p.

Matrices

Inverse of a matrix

Given an $N \times M$ matrix $\mathbf{P}$

if $\mathbf{PR} = \mathbf{I}_N$ then $\mathbf{R}$ is a *right inverse* of $\mathbf{P}$; this exists if and only if $\mathbf{P}$ has rank N.

$$\mathbf{P}^{-1} = \frac{\text{Adj } \mathbf{P}}{\det \mathbf{P}} \qquad [\text{Adj } \mathbf{P}]_{km} = \text{cofactor of } P_{mk}.$$

$$\mathbf{P} = \mathbf{AB} \qquad \mathbf{P}^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1},$$

Eigenvalues and eigenvectors

the nonzero vector $\mathbf{v}$ is an **eigenvector** of the $M \times M$ matrix $\mathbf{P}$ if

$$\mathbf{Pv} = \lambda \mathbf{v}$$

The scalar λ is the **eigenvalue**

Matrices

Eigenvalues and eigenvectors

the nonzero vector $\mathbf{v}$ is an **eigenvector** of the $M \times M$ matrix $\mathbf{P}$ if

$$\mathbf{P}\mathbf{v} = \lambda \mathbf{v}$$

The scalar λ is the **eigenvalue**

characteristic equation $\quad \det\left[s\mathbf{I} - \mathbf{P}\right] = 0.$

The determinant and trace of an $M \times M$ matrix $\mathbf{P}$ are related to its M eigenvalues λ_k as follows:

$$\det \mathbf{P} = \prod_{k=0}^{M-1} \lambda_k \quad \text{and} \quad \mathrm{Tr}\left(\mathbf{P}\right) = \sum_{k=0}^{M-1} \lambda_k .$$

Matrices

Eigenvalues and eigenvectors

$\mathbf{P}$ has an eigenvalue equal to zero if and only if it is singular (determinant equal to zero).

$\mathbf{P}$ and $\mathbf{P}^T$ have the same set of eigenvalues including multiplicity.

For a (lower or upper) triangular matrix, the eigenvalues are equal to the diagonal elements. Diagonal matrices also have this property.

For nonsingular $\mathbf{P}$, the eigenvalues of $\mathbf{P}^{-1}$ are reciprocals of those of $\mathbf{P}$.

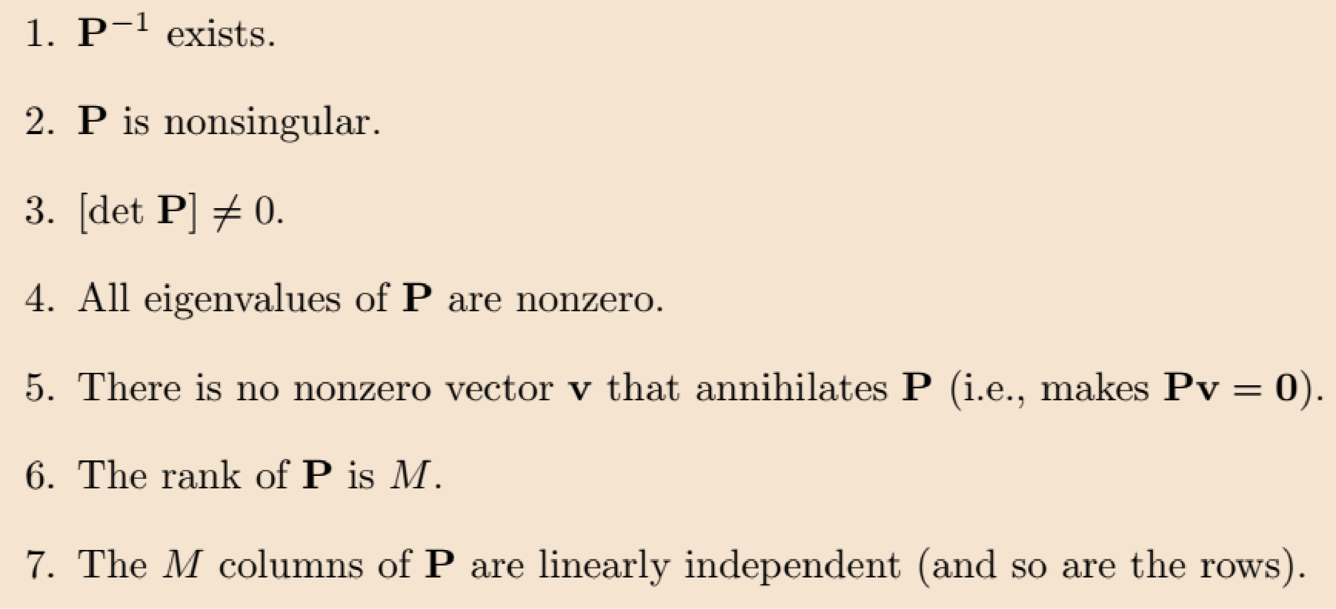
If λ_k are the eigenvalues of $\mathbf{P}$, the eigenvalues of $\mathbf{P} + \sigma\mathbf{I}$ are $\lambda_k + \sigma$.

Matrices

Eigenvalues and eigenvectors

The matrix $\mathbf{T}^{-1}\mathbf{PT}$ has the same set of eigenvalues as $\mathbf{P}$ (including multiplicity). This is true for any nonsingular $\mathbf{T}$. The matrix $\mathbf{T}^{-1}\mathbf{PT}$ is said to be a **similarity transformation** of $\mathbf{P}$.

Equivalent Statements:



1. $\mathbf{P}^{-1}$ exists.
2. $\mathbf{P}$ is nonsingular.
3. $[\det \mathbf{P}] \neq 0$.
4. All eigenvalues of $\mathbf{P}$ are nonzero.
5. There is no nonzero vector $\mathbf{v}$ that annihilates $\mathbf{P}$ (i.e., makes $\mathbf{Pv} = \mathbf{0}$).
6. The rank of $\mathbf{P}$ is M.
7. The M columns of $\mathbf{P}$ are linearly independent (and so are the rows).

Matrices

Matrix inversion lemma

$$(\mathbf{P}+\mathbf{QRS})^{-1}=\mathbf{P}^{-1}-\mathbf{P}^{-1}\mathbf{Q}(\mathbf{SP}^{-1}\mathbf{Q}+\mathbf{R}^{-1})^{-1}\mathbf{SP}^{-1}.$$

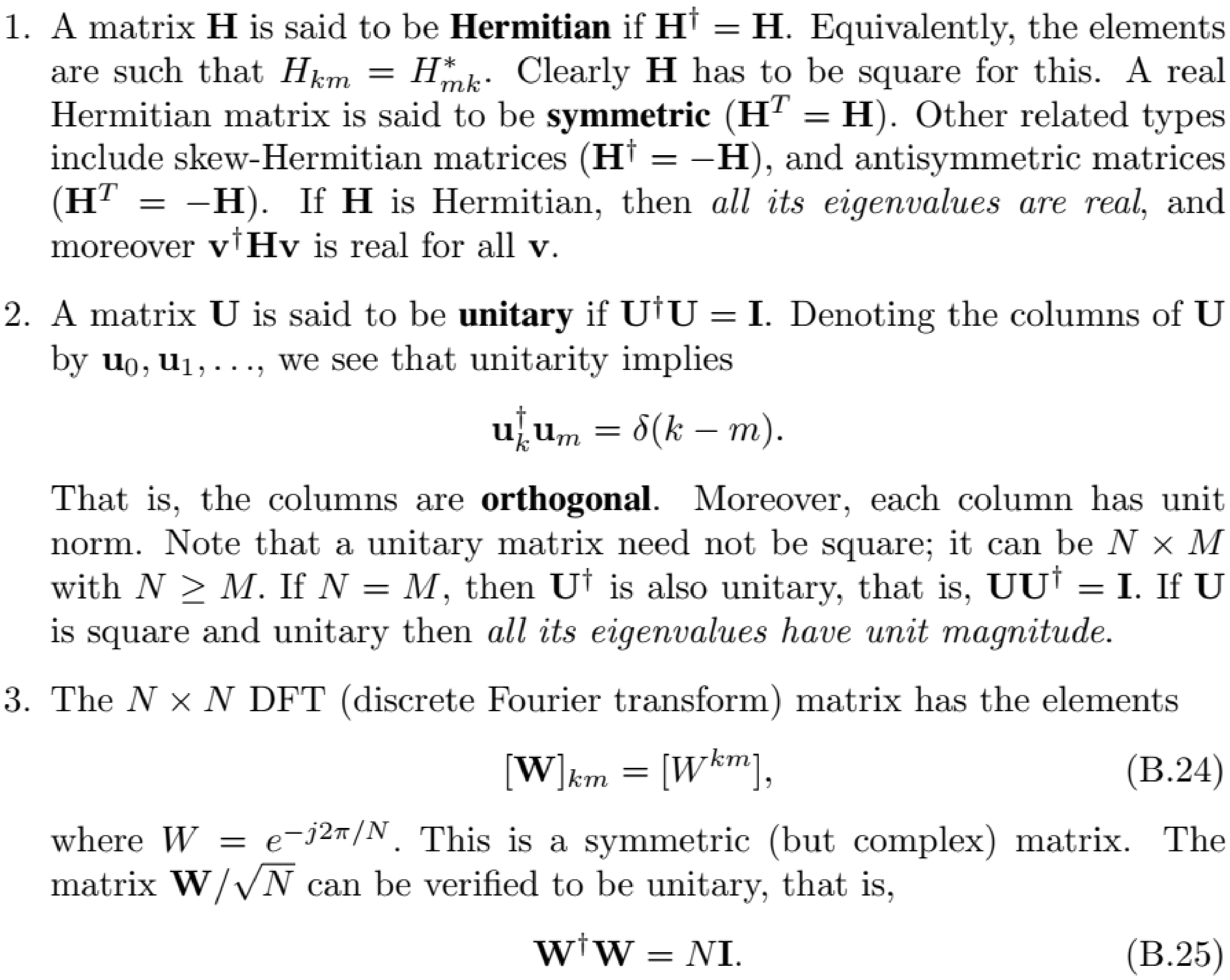
Partitioned matrices

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}=\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{CA}^{-1} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{A} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}-\mathbf{CA}^{-1}\mathbf{B}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{A}^{-1}\mathbf{B}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}$$

$$\det\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}=\det\mathbf{A}\times\ \det\left(\mathbf{D}-\mathbf{CA}^{-1}\mathbf{B}\right).$$

Matrices

Matrices with special properties



1. A matrix $\mathbf{H}$ is said to be **Hermitian** if $\mathbf{H}^\dagger = \mathbf{H}$. Equivalently, the elements are such that $H_{km} = H^*_{mk}$. Clearly $\mathbf{H}$ has to be square for this. A real Hermitian matrix is said to be **symmetric** ($\mathbf{H}^T = \mathbf{H}$). Other related types include skew-Hermitian matrices ($\mathbf{H}^\dagger = -\mathbf{H}$), and antisymmetric matrices ($\mathbf{H}^T = -\mathbf{H}$). If $\mathbf{H}$ is Hermitian, then *all its eigenvalues are real*, and moreover $\mathbf{v}^\dagger\mathbf{H}\mathbf{v}$ is real for all $\mathbf{v}$.

2. A matrix $\mathbf{U}$ is said to be **unitary** if $\mathbf{U}^\dagger\mathbf{U} = \mathbf{I}$. Denoting the columns of $\mathbf{U}$ by $\mathbf{u}_0, \mathbf{u}_1, \ldots$, we see that unitarity implies

$$\mathbf{u}_k^\dagger \mathbf{u}_m = \delta(k-m).$$

   That is, the columns are **orthogonal**. Moreover, each column has unit norm. Note that a unitary matrix need not be square; it can be $N \times M$ with $N \geq M$. If $N = M$, then $\mathbf{U}^\dagger$ is also unitary, that is, $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$. If $\mathbf{U}$ is square and unitary then *all its eigenvalues have unit magnitude.*

3. The $N \times N$ DFT (discrete Fourier transform) matrix has the elements

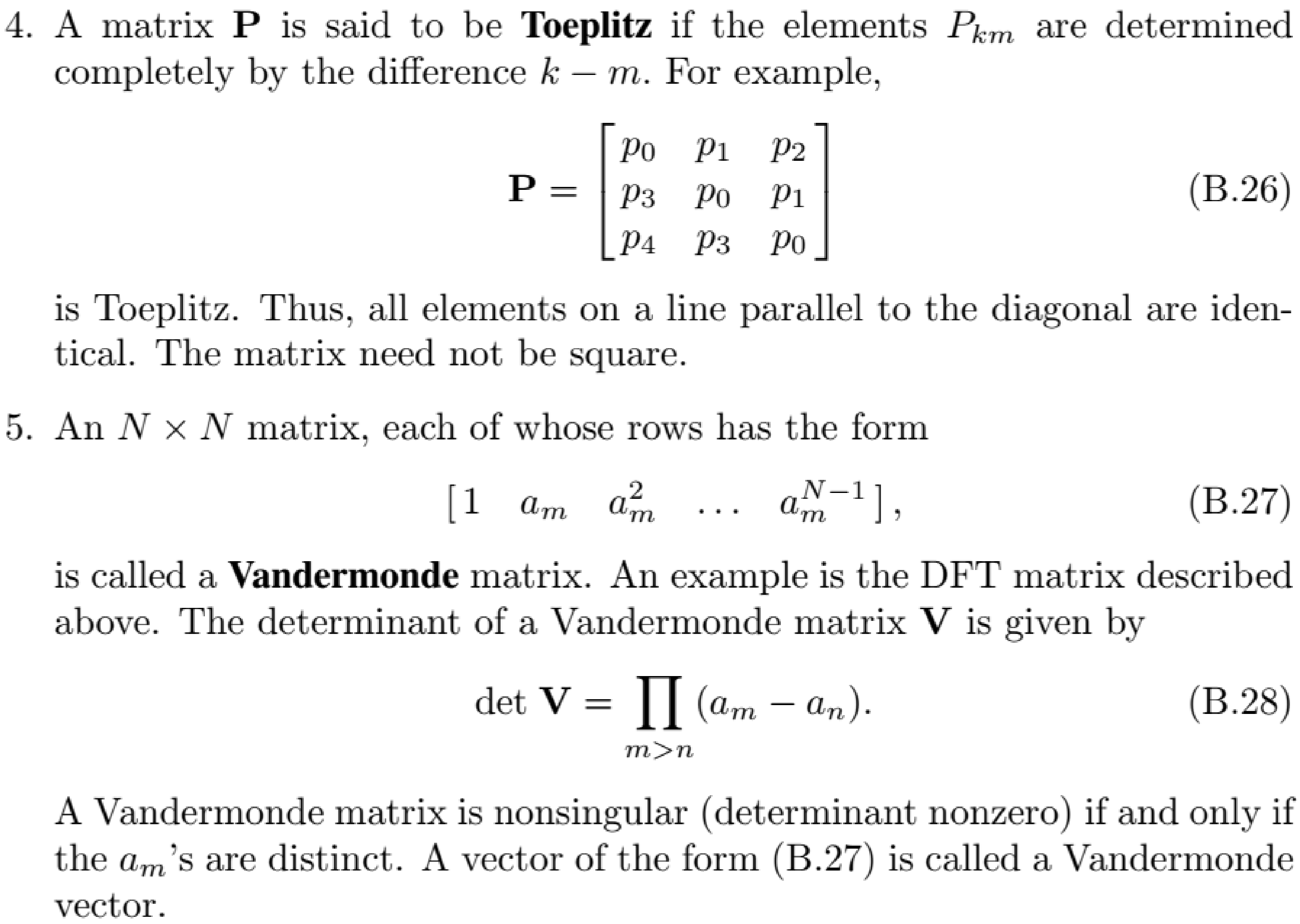
$$[\mathbf{W}]_{km} = [W^{km}], \qquad \text{(B.24)}$$

   where $W = e^{-j2\pi/N}$. This is a symmetric (but complex) matrix. The matrix $\mathbf{W}/\sqrt{N}$ can be verified to be unitary, that is,

$$\mathbf{W}^\dagger\mathbf{W} = N\mathbf{I}. \qquad \text{(B.25)}$$

Matrices

Matrices with special properties



4. A matrix $\mathbf{P}$ is said to be **Toeplitz** if the elements P_{km} are determined completely by the difference $k - m$. For example,

$$\mathbf{P} = \begin{bmatrix} p_0 & p_1 & p_2 \\ p_3 & p_0 & p_1 \\ p_4 & p_3 & p_0 \end{bmatrix} \tag{B.26}$$

is Toeplitz. Thus, all elements on a line parallel to the diagonal are identical. The matrix need not be square.

5. An $N \times N$ matrix, each of whose rows has the form

$$[1 \quad a_m \quad a_m^2 \quad \ldots \quad a_m^{N-1}], \tag{B.27}$$

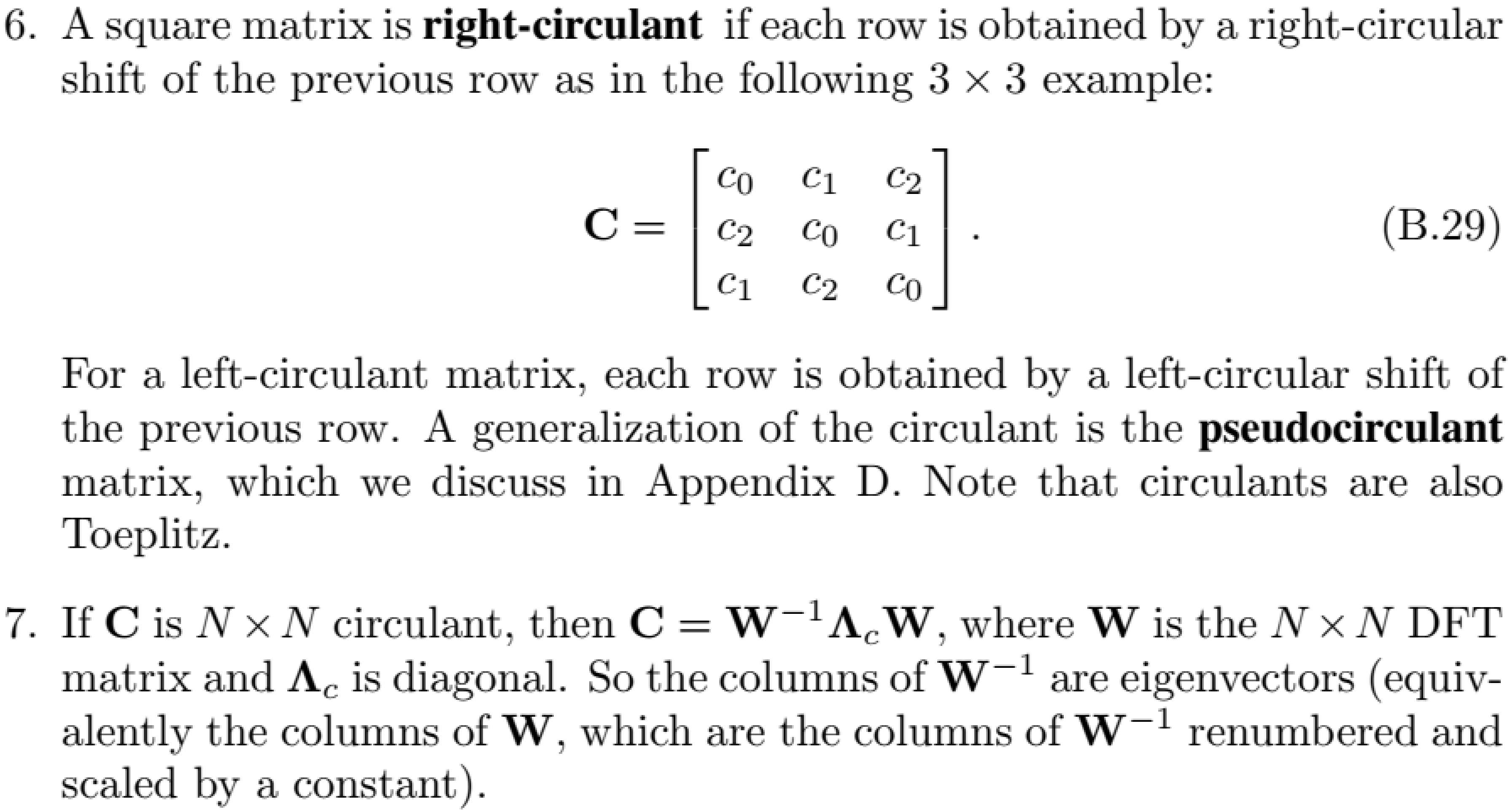
is called a **Vandermonde** matrix. An example is the DFT matrix described above. The determinant of a Vandermonde matrix $\mathbf{V}$ is given by

$$\det \mathbf{V} = \prod_{m>n} (a_m - a_n). \tag{B.28}$$

A Vandermonde matrix is nonsingular (determinant nonzero) if and only if the a_m's are distinct. A vector of the form (B.27) is called a Vandermonde vector.

Matrices

Matrices with special properties



6. A square matrix is **right-circulant** if each row is obtained by a right-circular shift of the previous row as in the following 3 × 3 example:

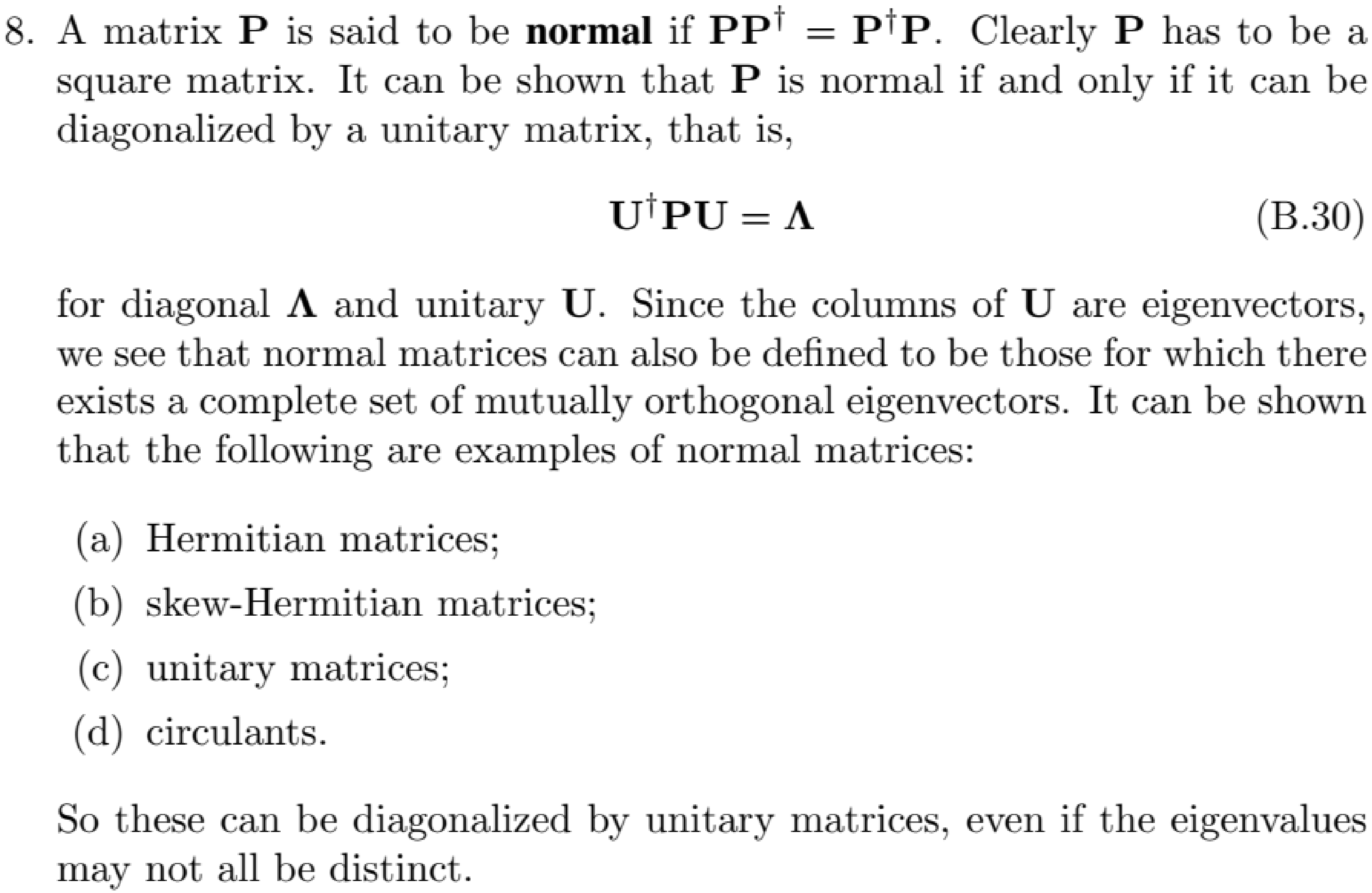
$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 & c_2 \\ c_2 & c_0 & c_1 \\ c_1 & c_2 & c_0 \end{bmatrix}. \tag{B.29}$$

For a left-circulant matrix, each row is obtained by a left-circular shift of the previous row. A generalization of the circulant is the **pseudocirculant** matrix, which we discuss in Appendix D. Note that circulants are also Toeplitz.

7. If $\mathbf{C}$ is $N \times N$ circulant, then $\mathbf{C} = \mathbf{W}^{-1}\mathbf{\Lambda}_c\mathbf{W}$, where $\mathbf{W}$ is the $N \times N$ DFT matrix and $\mathbf{\Lambda}_c$ is diagonal. So the columns of $\mathbf{W}^{-1}$ are eigenvectors (equivalently the columns of $\mathbf{W}$, which are the columns of $\mathbf{W}^{-1}$ renumbered and scaled by a constant).

Matrices

Matrices with special properties



8. A matrix $\mathbf{P}$ is said to be **normal** if $\mathbf{P}\mathbf{P}^\dagger = \mathbf{P}^\dagger\mathbf{P}$. Clearly $\mathbf{P}$ has to be a square matrix. It can be shown that $\mathbf{P}$ is normal if and only if it can be diagonalized by a unitary matrix, that is,

$$\mathbf{U}^\dagger\mathbf{P}\mathbf{U} = \mathbf{\Lambda} \tag{B.30}$$

for diagonal $\mathbf{\Lambda}$ and unitary $\mathbf{U}$. Since the columns of $\mathbf{U}$ are eigenvectors, we see that normal matrices can also be defined to be those for which there exists a complete set of mutually orthogonal eigenvectors. It can be shown that the following are examples of normal matrices:

(a) Hermitian matrices;

(b) skew-Hermitian matrices;

(c) unitary matrices;

(d) circulants.

So these can be diagonalized by unitary matrices, even if the eigenvalues may not all be distinct.

Matrices

Positive definite matrices

$\mathbf{v}^\dagger \mathbf{P} \mathbf{v} > 0$

For any $N \times N$ matrix $\mathbf{P}$, the scalar

$$\phi = \mathbf{v}^\dagger \mathbf{P} \mathbf{v} \tag{B.35}$$

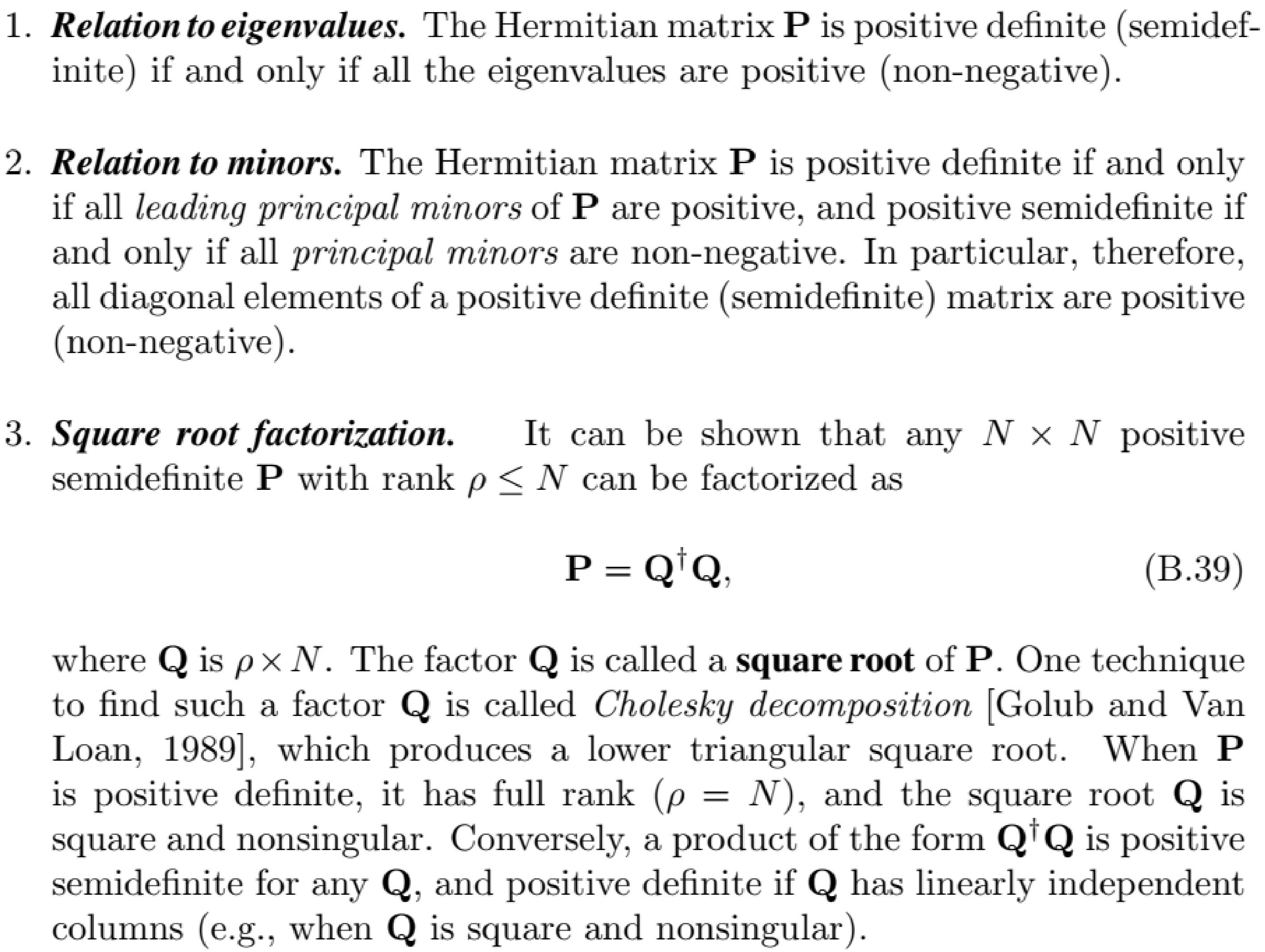
is said to be a **quadratic form**, where $\mathbf{v}$ is a column vector. When $\mathbf{P}$ is Hermitian, $\mathbf{v}^\dagger \mathbf{P} \mathbf{v}$ is guaranteed to be real. If the Hermitian matrix $\mathbf{P}$ is such that $\mathbf{v}^\dagger \mathbf{P} \mathbf{v} > 0$ for $\mathbf{v} \neq \mathbf{0}$, we say that $\mathbf{P}$ is positive definite. If $\mathbf{v}^\dagger \mathbf{P} \mathbf{v} \geq 0$ for all $\mathbf{v}$, then $\mathbf{P}$ is positive semidefinite. Negative definiteness and semidefiniteness are similarly defined. If $\mathbf{P}$ is positive definite, we write it as

$$\mathbf{P} > \mathbf{0}, \tag{B.36}$$

Matrices

Positive definite matrices

$\mathbf{v}^\dagger \mathbf{P} \mathbf{v} > 0$



1. ***Relation to eigenvalues.*** The Hermitian matrix $\mathbf{P}$ is positive definite (semidefinite) if and only if all the eigenvalues are positive (non-negative).

2. ***Relation to minors.*** The Hermitian matrix $\mathbf{P}$ is positive definite if and only if all *leading principal minors* of $\mathbf{P}$ are positive, and positive semidefinite if and only if all *principal minors* are non-negative. In particular, therefore, all diagonal elements of a positive definite (semidefinite) matrix are positive (non-negative).

3. ***Square root factorization.*** It can be shown that any $N \times N$ positive semidefinite $\mathbf{P}$ with rank $\rho \leq N$ can be factorized as

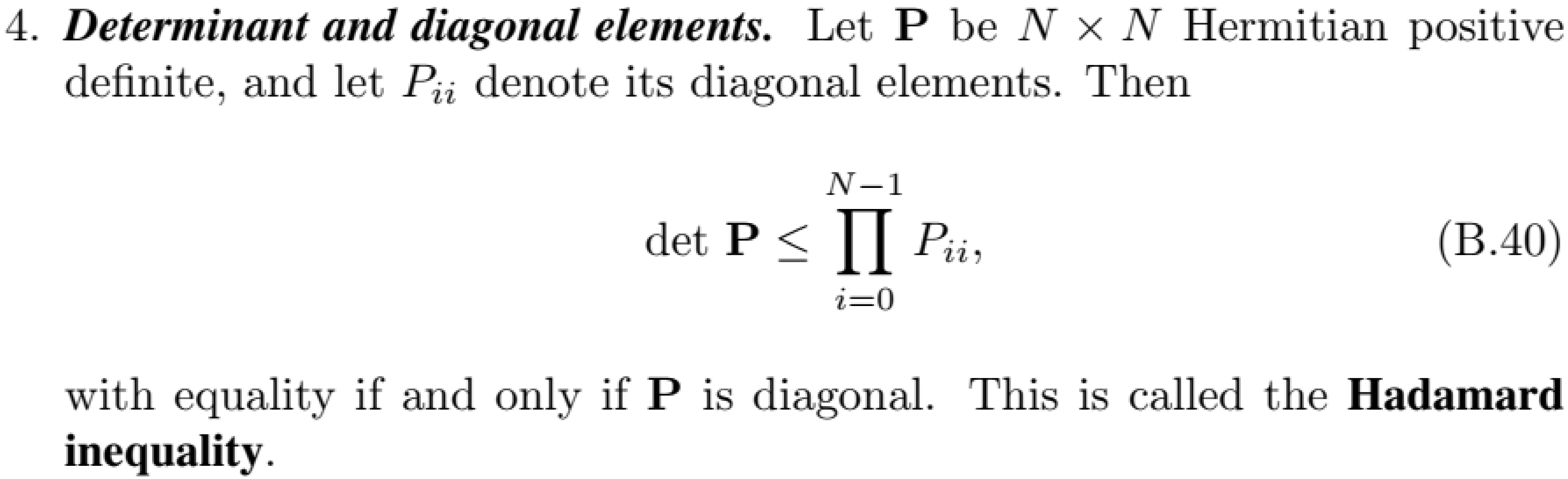
$$\mathbf{P} = \mathbf{Q}^\dagger \mathbf{Q}, \tag{B.39}$$

where $\mathbf{Q}$ is $\rho \times N$. The factor $\mathbf{Q}$ is called a **square root** of $\mathbf{P}$. One technique to find such a factor $\mathbf{Q}$ is called *Cholesky decomposition* [Golub and Van Loan, 1989], which produces a lower triangular square root. When $\mathbf{P}$ is positive definite, it has full rank ($\rho = N$), and the square root $\mathbf{Q}$ is square and nonsingular. Conversely, a product of the form $\mathbf{Q}^\dagger \mathbf{Q}$ is positive semidefinite for any $\mathbf{Q}$, and positive definite if $\mathbf{Q}$ has linearly independent columns (e.g., when $\mathbf{Q}$ is square and nonsingular).

Matrices

Positive definite matrices

$\mathbf{v}^{\dagger}\mathbf{P}\mathbf{v} > 0$



4. ***Determinant and diagonal elements.*** Let $\mathbf{P}$ be $N \times N$ Hermitian positive definite, and let P_{ii} denote its diagonal elements. Then

$$\det \mathbf{P} \leq \prod_{i=0}^{N-1} P_{ii}, \tag{B.40}$$

with equality if and only if $\mathbf{P}$ is diagonal. This is called the **Hadamard inequality**.

Matrices

Rayleigh-Ritz principle

Let $\mathbf{P}$ be $N \times N$ Hermitian smallest and largest eigenvalues,

$$\max_{\mathbf{v}^\dagger \mathbf{v}=1} \mathbf{v}^\dagger \mathbf{P} \mathbf{v} = \lambda_{max}.$$

$$\min_{\mathbf{v}^\dagger \mathbf{v}=1} \mathbf{v}^\dagger \mathbf{P} \mathbf{v} = \lambda_{min}.$$

Singular value decomposition

$$\mathbf{A} = \underbrace{\mathbf{U}}_{P\times P}\underbrace{\mathbf{S}}_{P\times M}\underbrace{\mathbf{V}^{\dagger}}_{M\times M},$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, that is,

$$\mathbf{U}^{\dagger}\mathbf{U} = \mathbf{U}\mathbf{U}^{\dagger} = \mathbf{I}_P, \qquad \mathbf{V}\mathbf{V}^{\dagger} = \mathbf{V}^{\dagger}\mathbf{V} = \mathbf{I}_M,$$

and $\mathbf{S}$ is a diagonal matrix with real non-negative diagonal elements $\sigma_k \geq 0$.

The diagonal elements σ_k are called the **singular values** of $\mathbf{A}$.

Left inverse computed from SVD

$$\mathbf{A}^{\#} = \mathbf{V}\,[\,\mathbf{\Sigma}^{-1} \quad \mathbf{0}\,]\,\mathbf{U}^{\dagger}.$$

$$\mathbf{A}^{\#} = (\mathbf{A}^{\dagger}\mathbf{A})^{-1}\mathbf{A}^{\dagger}$$

$$\mathbf{A}^{\#} = \mathbf{A}^{\dagger}(\mathbf{A}\mathbf{A}^{\dagger})^{-1}.$$

Singular value decomposition

Frobenius norm and SVD

$$\|\mathbf{A}\|^2 = \sum_{k=0}^{P-1} \sum_{m=0}^{M-1} |a_{km}|^2.$$

$$\|\mathbf{A}\|^2 = \mathrm{Tr}(\mathbf{A}^\dagger \mathbf{A}) = \mathrm{Tr}(\mathbf{A}\mathbf{A}^\dagger),$$

$$\|\mathbf{A}\|^2 = \sum_{k=0}^{M-1} \sigma_k^2.$$