http://elec3004.com

FFT
— then →
Stochastic Processes: Markov Chains

ELEC 3004: **Systems**: Signals & Controls
Tim Sherry & Surya Singh

Lecture 20: Part II

elec3004@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~elec3004/

May 14, 2019

FFT

Example: 8-point DFT Matrix

x(0) x(2) x(4) x(6) Even samples

$$
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}
=
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\
W_8^0 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\
W_8^0 & W_8^2 & W_8^4 & W_8^6 & W_8^0 & W_8^2 & W_8^4 & W_8^6 \\
W_8^0 & W_8^3 & W_8^6 & W_8^1 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\
W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 \\
W_8^0 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8^1 & W_8^6 & W_8^3 \\
W_8^0 & W_8^6 & W_8^4 & W_8^2 & W_8^0 & W_8^6 & W_8^4 & W_8^2 \\
W_8^0 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8^1
\end{bmatrix}
\cdot
\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}
$$

Repeated complex multiplications in EVEN rows

Re-ordered DFT Matrix

Separate even and odd row operations (and re-order input vector)

$$
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}
=
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\
W_8^0 & W_8^4 & W_8^2 & W_8^6 & W_8^1 & W_8^5 & W_8^3 & W_8^7 \\
W_8^0 & W_8^0 & W_8^4 & W_8^4 & W_8^2 & W_8^2 & W_8^6 & W_8^6 \\
W_8^0 & W_8^4 & W_8^6 & W_8^2 & W_8^3 & W_8^7 & W_8^1 & W_8^5 \\
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^4 & W_8^4 & W_8^4 & W_8^4 \\
W_8^0 & W_8^4 & W_8^2 & W_8^6 & W_8^5 & W_8^1 & W_8^7 & W_8^3 \\
W_8^0 & W_8^0 & W_8^4 & W_8^4 & W_8^6 & W_8^6 & W_8^2 & W_8^2 \\
W_8^0 & W_8^4 & W_8^6 & W_8^2 & W_8^7 & W_8^3 & W_8^5 & W_8^1
\end{bmatrix}
\cdot
\begin{bmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \\ x(1) \\ x(5) \\ x(3) \\ x(7) \end{bmatrix}
$$

Even samples   Odd samples

Phasor Rotational Symmetry

To highlight repeated computations on odd samples
as $W_8^4 = -W_8^0$, $W_8^5 = -W_8^1$, $W_8^6 = -W_8^2$, $W_8^7 = -W_8^3$

$$
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}
=
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & W_8^1 & -W_8^1 & W_8^3 & -W_8^3 \\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & W_8^2 & W_8^2 & -W_8^2 & -W_8^2 \\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & W_8^3 & -W_8^3 & W_8^1 & -W_8^1 \\
W_8^0 & W_8^0 & W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & -W_8^0 & -W_8^0 \\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & -W_8^1 & W_8^1 & -W_8^3 & W_8^3 \\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & -W_8^2 & -W_8^2 & W_8^2 & W_8^2 \\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & -W_8^3 & W_8^3 & -W_8^1 & W_8^1
\end{bmatrix}
\cdot
\begin{bmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \\ x(1) \\ x(5) \\ x(3) \\ x(7) \end{bmatrix}
$$

Upper & lower left-hand quarters are identical
Right hand quarters identical except sign difference!

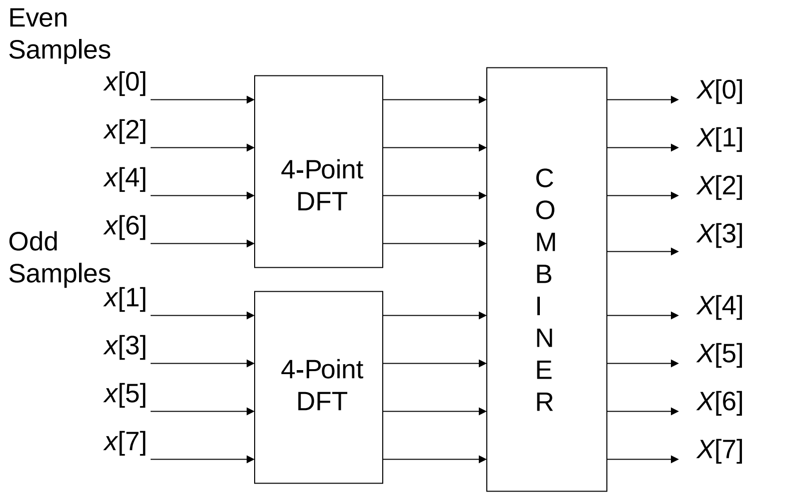
Adding "Twiddle Factors"

$$
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \times W_8^0 & W_8^0 \times W_8^0 & W_8^0 \times W_8^0 & W_8^0 \times W_8^0 \\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & W_8^1 \times W_8^0 & W_8^1 \times -W_8^0 & W_8^1 \times W_8^2 & W_8^1 \times -W_8^2 \\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & W_8^2 \times W_8^0 & W_8^2 \times W_8^0 & W_8^2 \times -W_8^0 & W_8^2 \times -W_8^0 \\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & W_8^3 \times W_8^0 & W_8^3 \times -W_8^0 & W_8^3 \times -W_8^2 & W_8^3 \times W_8^2 \\
W_8^0 & W_8^0 & W_8^0 & W_8^0 & -W_8^0 \times W_8^0 & -W_8^0 \times W_8^0 & -W_8^0 \times W_8^0 & -W_8^0 \times W_8^0 \\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & -W_8^1 \times W_8^0 & -W_8^1 \times -W_8^0 & -W_8^1 \times W_8^2 & -W_8^1 \times -W_8^2 \\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & -W_8^2 \times W_8^0 & -W_8^2 \times W_8^0 & -W_8^2 \times -W_8^0 & -W_8^2 \times -W_8^0 \\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & -W_8^3 \times W_8^0 & -W_8^3 \times -W_8^0 & -W_8^3 \times -W_8^2 & -W_8^3 \times W_8^2
\end{bmatrix}
$$

i.e., 8-point DFT reduced to two 4-point DFT's only need calculate upper left and right quarters

Twiddle Factors make the left and right hand quarters identical

8-Point DFT as Two 4-Point DFTs



Combiner adds twiddle factors to data

Radix-2 FFT

Each 4-point DFT can be reduced to two 2-point DFT's

$$\begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & -W^0 & W^2 & -W^2 \\ W^0 & W^0 & -W^0 & -W^0 \\ W^0 & -W^0 & -W^2 & W^2 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 \times W^0 & W^0 \times W^0 \\ W^0 & -W^0 & W^2 \times W^0 & W^2 \times -W^0 \\ W^0 & W^0 & -W^0 \times W^0 & -W^0 \times W^0 \\ W^0 & -W^0 & -W^2 \times W^0 & -W^2 \times -W^0 \end{bmatrix}$$

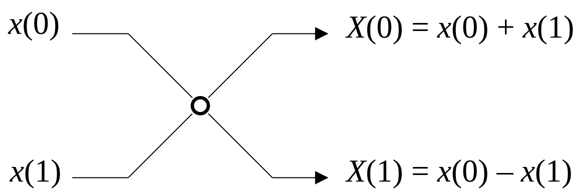
2x2 Quadrants are identical (with twiddle factors)

Two-point "Butterfly" operation

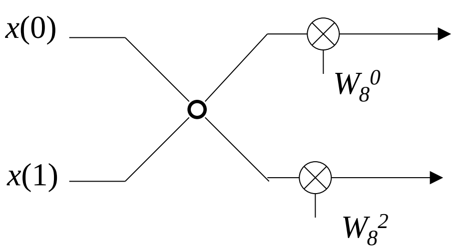
$$\begin{bmatrix} X(0) \\ X(1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 \\ W^0 & -W^0 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \end{bmatrix}$$

$$\begin{bmatrix} X(0) \\ X(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \end{bmatrix}$$

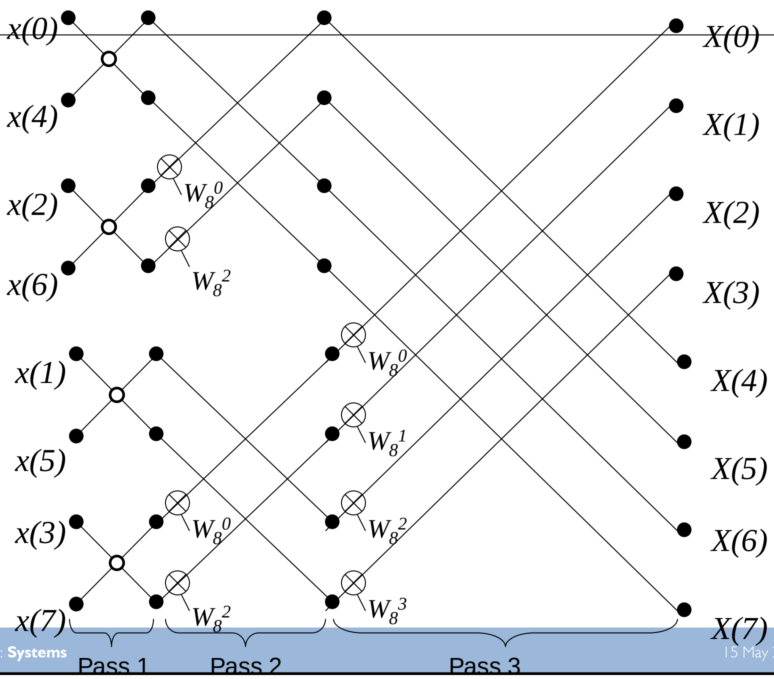
Two Point Butterfly



With twiddle factors:



8-point radix-2 DIT FFT flowgraph:

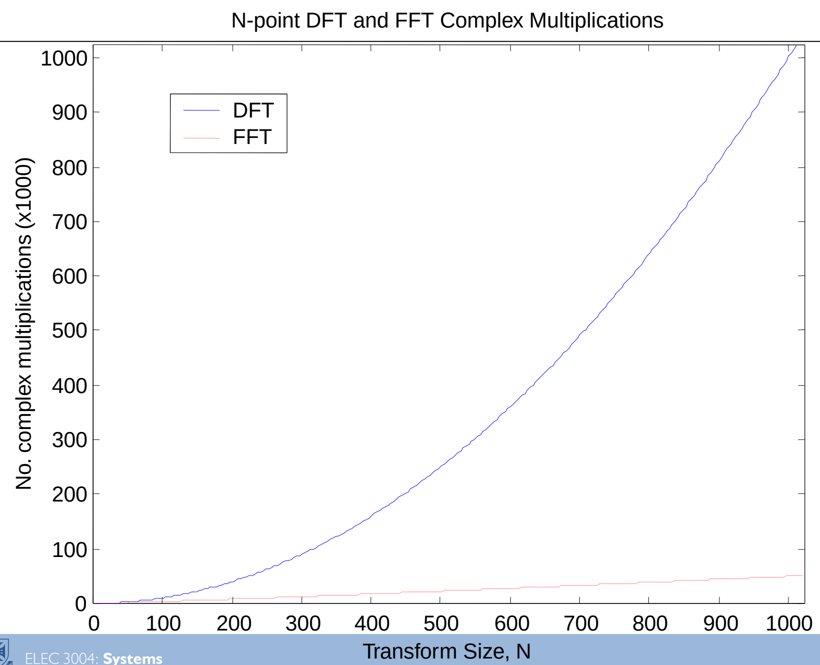


Features of the FFT

- Reduce complex multiplications from N^2 to:
  - $\left(\frac{N}{2}\right)\log_2(N)$
  - As there are $\log_2(N)$ passes
  - Each pass requires $\frac{N}{2}$ complex multiplications

- Disadvantages
  - More complex memory addressing
    - To get appropriate samples pairs for each butterfly
  - FFT can be slower (than DFT) for small N (< 16)

- What about the IFFT? We can use same FFT algorithm
  - change sign of twiddle factors
  - and scale output to get $x[n]$



What does it let us do

- For high performance applications, the FFT can be the difference between feasible and infeasible
- EG – 4K video:
  - There are ~8M pixels
  - $N^2 = 6.4 \cdot 10^{13}$ $\quad N \cdot Log_2(N) = 1.6 \cdot 10^8$
  - You need to do this 30x per second
  - So the flops is on the order of $2 \cdot 10^{15}$ = 2 PFlops using the DFT vs $4 \cdot 10^9$=4 Gflops using the FFT
- An Nvidia V100 maxes out at 120TFlops, a standard CPU is about 100GFlops

Alternative FFT Algorithms

- Only case covered so far is
  - (one case of) radix-2 decimation in time (DIT) FFT
  - requires sequence length, N, to be a power of 2
  - achieved by 'zero padding' sequence to desired, N

- Decimation in Frequency
  - similar to DIT, twiddle factors on outputs

- Alternatives to radix-2 decomposition
  - Radix 3: for sequence length, N = power of 3
  - Radix 4: twice as fast as radix 2 FFT
    - half number of passes, log4(N)
  - Split radix: mixtures of the above

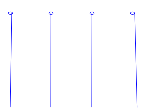
Applications of the FFT

- Fast (circular) Convolution
  - Convolution requires N^2 MAC operations ☹
  - more efficient alternative via the FFT ☺
    - Take FFT of both sequences
    - Multiply them together (point-wise)
    - Take IFFT to get the result

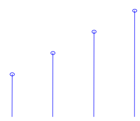
      ["Hello FFT-W! *Bonjour cuFFT!*"]
  - Zeropad and you have linear convolution
- Spectral Analysis
  - Estimate (power) spectrum with less computations
  - i.e., what frequencies in our signal are carrying power (i.e., carrying information) ?
- Fast Cross-correlation
  - E.g., correlation detector in digital comm's

(Linear) Convolution

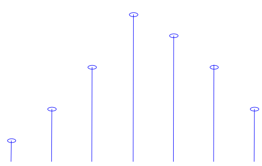
$h[n]$ = {1 1 1 1}



$x[n]$ = {0.5 0.75 1.0 1.25}



$y[n] = x[n]*h[n]$ = {0.5 1.25 2.25 3.5 3.0 2.25 1.25}



In general: length($y[n]$) = length($x[n]$) + length($h[n]$) - 1

Circular Convolution

Given $X[k]$ = DFT{$x[n]$} and $H[k]$ = DFT{$h[n]$}

from convolution theorem we know

$$\text{IDFT}\{X[k]\cdot H[k]\} \equiv x[n]*h[n]$$

$$\text{IDFT}\{X[k]\cdot H[k]\} = \{3.5\ 3.5\ 3.5\ 3.5\} \leftarrow \text{Wrong Length!}$$

Solution: zero pad both sequences to required length

$h_p[n]$ = {1 1 1 1 0 0 0 } $x_p[n]$ = {0.5 0.75 1.0 1.25 0 0 0}

$$\text{IDFT}\{X_p[k]\cdot H_p[k]\} = [0.5\ 1.25\ \ 2.25\ \ 3.5\ \ 3.0\ \ 2.25\ \ 1.25]$$

i.e., $x[n]$ and $h[n]$
are periodic in time

Spectral Analysis

- Power Spectral Density (PSD) defined as
  - Fourier Transform of Autocorrelation function

$$S_{xx}(w) = \sum_{m=-\infty}^{\infty} \varphi_{xx}(m)\exp(-jwm\Delta t)$$

- In practice, we estimate $S_{xx}(w)$ from $\{x[n]\}_0^{N-1}$
  - i.e., a finite length of sampled data
- This can be done using N - point DFT
  - and implemented using the FFT algorithm

Spectral Analysis

- Estimate of PSD is given by

$$\hat{S}_{xx}[k] = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n] \exp\left(\frac{-jnk2\pi}{N}\right)\right|^2$$

- This is known as a **periodogram**
  - DFT effectively implements narrow-band filter bank
  - calculate power (i.e., square) at each frequency *k*

- Again, window functions often required
  - to improve PSD estimate
  - e.g., Hanning, ~~Hamming~~, Bartlet etc

Spectral Analysis

- When finding PSD as DFT of $\phi^{xx[m]}$:
  - $\phi^{xx[m]}$ has an odd length! $(2M + 1)$

- Therefore, to use the radix-2 FFT we need to
  - zero pad $\phi^{xx[m]}$ to length = power of 2

- e.g., for $M = 2$, $\phi^{xx[m]}$ is of length 5
  - we need to zero pad to length 8, i.e.,
  - $\{\phi^{xx[-2]}\ \ \phi^{xx[-1]}\ \ \phi^{xx[0]}\ \ \phi^{xx[1]}\ \ \phi^{xx[2]}\ \ 0\ \ 0\ \ 0\}$
  - Note, sequence made causal (no change to PSD)

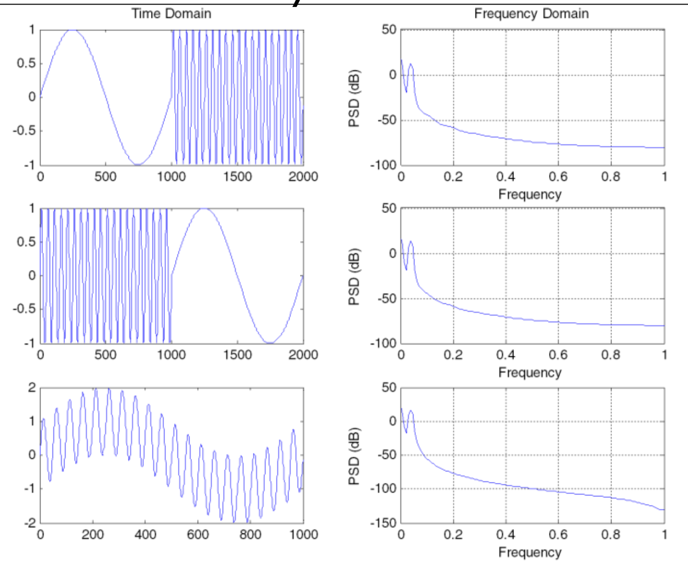
- This estimate of PSD is known as correlogram
  - Note, periodogram is most common estimate of PSD

Limitations of Fourier Analysis

Low – high

High – low

Mixed



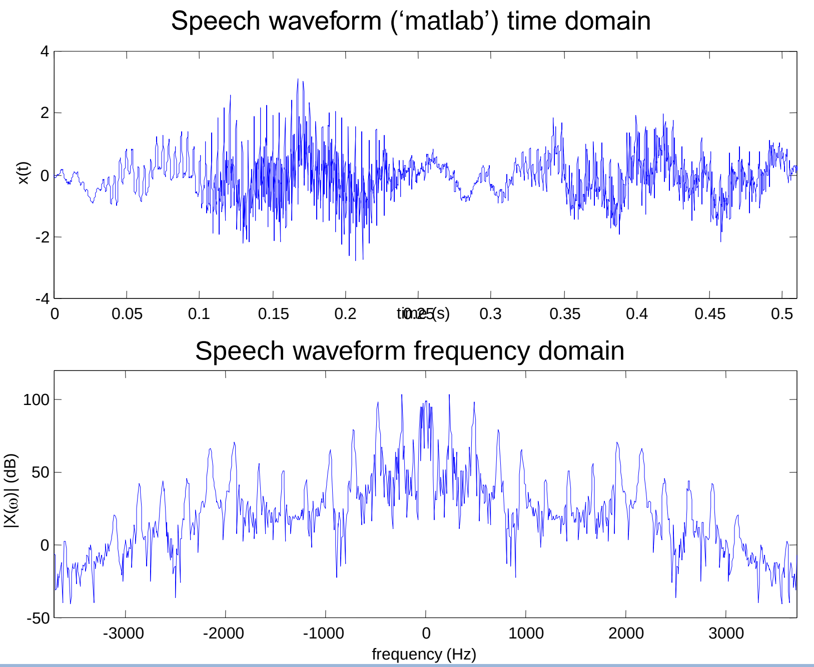
Note: These signals differ in Phase. PSD is zero phase as $F\{\phi_{xx}(k)\}$ real & even

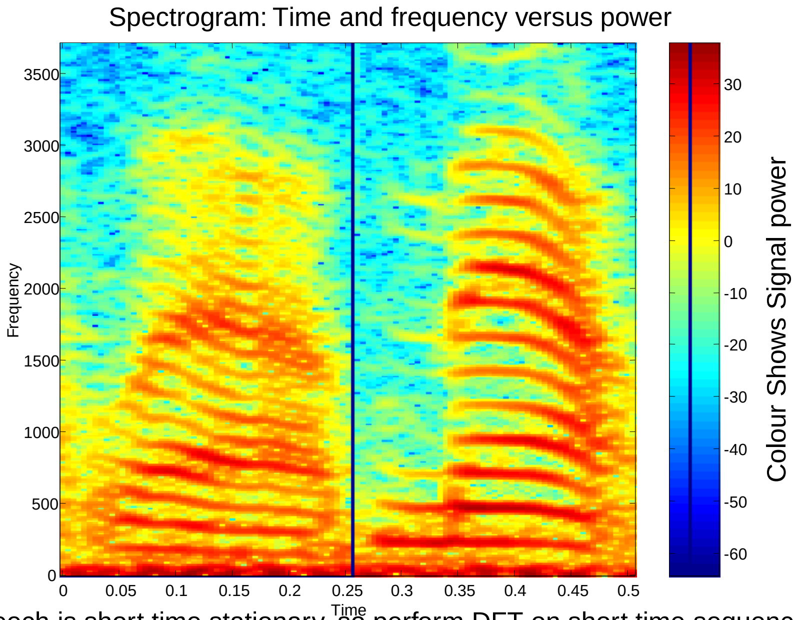
Time Frequency

(Mini-section)

Spectrum Analysis of Non-Stationary Signals

- Spectrum of non-deterministic Signal $X(w)$
  - is only valid if $x(t)$ is stationary
  - i.e., statistics of $x(t)$ do not change over time
- Real-world signals often only stationary over a short time period of time
  - e.g., speech: assumed stationary over t < 60ms
- Therefore, take 'short-time' DFT of signal
  - i.e., take multiple DFT's over stationary periods
  - plot how frequency components change over time
  - for speech the plot of time V frequency V power
    - is called a **Spectrogram**





Speech is short time stationary, so perform DFT on short time sequences

What about random processes?

- We have mentioned the concept of Gaussian noise on many occasions in the course, how do we treat this correctly?
- What about other random processes?
  - Laplacians
  - Bimodal distributions

Stochastic Processes

Deterministic Signal Processing

- The vast majority of the course has covered deterministic signal processing
  - Fourier
  - Laplace
  - LTI Systems
- However, most "interesting" processes are random (stochastic) in Nature
  - Communications
  - Biological signals
  - The perturbations acting on an aeroplane

Stochastic Signal Processing

- Correctly accounting for stochastic effects on a given dynamic system provides the potential for significant performance gains
- There are many tools designed for stochastic processes
- In Estimation:
  - The Kalman filter family
  - The Particle filter
  - EM, max Liklihood, Bayesian estimators (BLUE)
  - Markov process
- In Control:
  - The Linear Quadratic Regulator

What was the point of week 1:10 then?

- Stochastic estimation and control (often) uses or extends deterministic techniques in computing their results
- For example, the kalman filter operates by propagating the error and estimate through a deterministic system
- Often, a full stochastic treatment is unnecessary.

Markov Process

A random symbol generator

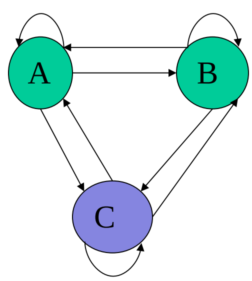
- We have discussed the concept of sampling regularly
- Till now, this sequence comprises of a set of integer values, which may have come from a stochastic process
- What about written language?
  - Consider sampling each character in turn
  - The sequence values are no longer numeric, they belong to one of 128(ascii) symbols
- Markov processes provide a convenient way to model sequences of randomly generated Symbolic data

Markov processes and English

- We would like to develop Sir Android Shakespeare (SAS), a virtual poet
- SAS needs to be able to generate ascii characters in a sequence which forms words, and which then forms sentences.
- Our intention is to train our SAS on the texts of William Shakespeare.

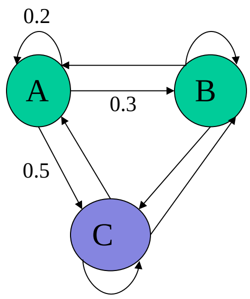
Markov processes and English

- You could consider building the system that is generating these symbols as a big state machine
- The next symbol is chosen probabilistically based on the current symbol



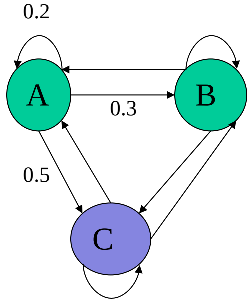
Probability of transition

- Probability of transition for A
- There are 9 possible transitions edges for a 3 node system



Rules on the probability of transition

- The sum of probabilities exiting each node must sum to one



Probability of transition as a matrix

- We can form a matrix which describes the probability of transition given a particular state described as a "one hot" vector

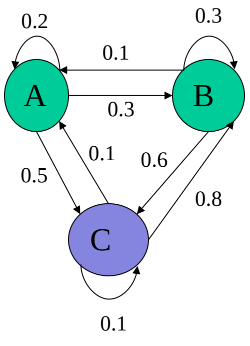
$$X_{-1} = \begin{matrix} 0 \\ 0 \\ 1 \end{matrix}$$

$$T_{X_1 \to X_0} = \begin{matrix} 0.2 & ? & ? \\ 0.3 & ? & ? \\ 0.5 & ? & ? \end{matrix}$$

$$P_{X_0|X_{-1}} = T_{X_{-1} \to X_0} \begin{matrix} 0 \\ 0 \\ 1 \end{matrix}$$

- The columns of T must sum to one (see previous slide)
- Note the transition matrix is often denoted as P

Lets fill in the rest of that transition Matrix



$$T_{X_0 \to X_{-1}} = \begin{matrix} 0.2 & 0.1 & 0.1 \\ 0.3 & 0.3 & 0.8 \\ 0.5 & 0.6 & 0.1 \end{matrix}$$

Now What?

- We can generate a sequence by sampling from the distribution $P_{X_0|X_{-1}}$

$$P_{X_T|X_{T-1}} = T_{X_0 \to X_{-1}} \cdot X_{T-1}$$

$$X_T \sim P_{X_T|X_{T-1}}$$

- We can estimate the overall distribution of symbols

Estimating the distribution of symbols

- We want to find the steady state distribution of X.
- Propagate the transition probability

$$P_{SS} = Lim_{N\to\infty}\left(T_{X_0 \to X_{-1}}^{\mathrm{N}} \cdot X_0\right)$$

- Or, more easily solve:

$$\mu = \mu \cdot T_{X_0 \to X_{-1}}$$

THAT'S AN EIGENVECTOR PROBLEM!

- **The distribution of the symbols are simply the left-eigenvectors of the transition matrix**

$$\mathrm{Eig}(P^T)$$

- To solve, you can solve the characteristic Eq.

$$|P^T - \lambda\, I| = 0$$

- We will go through how to solve the eq above in the tutes for a 2x2 matrix

Can we train the monkey to write Shakespeare?

- We can estimate $T_{X_0 \to X_{-1}}$ by inspection of the works of Shakespeare
- For each character in the work, look at X_t and X_{t-1}
- Add one to the corresponding element in our estimate $\hat{T}$
- Say we have the character pair "ac"

$$\hat{T} = \hat{T} + \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{matrix}$$

Can we train the monkey to write Shakespeare?

- Then normalise the columns

$$\hat{T} = \begin{matrix} 203 & 1001 & 10010 \\ 299 & 4002 & 80990 \\ 534 & 6000 & 9990 \end{matrix}$$

$$\hat{T} = \begin{matrix} 0.2003 & 0.1001 & 0.1001 \\ 0.2999 & 0.4002 & 0.8099 \\ 0.5034 & 0.6000 & 0.0999 \end{matrix}$$

- (In the real world case there are 128 possible Ascii characters, so the matrix is much bigger)

Does it work?

- So for a first order markov model we get….

t I amy, vin. id wht omanly heay atuss n macon aresethe hired boutwhe t, tl, ad torurest t plur I wit hengamind tarer-plarody thishand.

Higher order markov models

- What if we consider a higher order model
- $X_T \sim P_{X_T|X_{T-1},X_{T-2}\dots X_{T-N}}$

- We can form a 2nd order Markov model, where we consider a new set of symbols $U = [X_1\ X_2]$
- The Symbol U has K^2 Unique combinations, where K is the number of symbols present in X
- There are still only K unique transitions for each state
- The transition matrix is therefore size $[K\ K^N]$

Does that work?

- 2nd order
- **Ther I the heingoind of-pleat, blur it dwere wing waske hat trooss. Yout lar on wassing, an sit." "Yould," "I that vide was nots ther.**

- **3rd**
- **I has them the saw the secorrow. And wintails on my my ent, thinks, fore voyager lanated the been elsed helder was of him a very free bottlemarkable,**

- **4th**
- **His heard." "Exactly he very glad trouble, and by Hopkins! That it on of the who difficentralia. He rushed likely?" "Blood night that.**

- https://blog.codinghorror.com/markov-and-you/

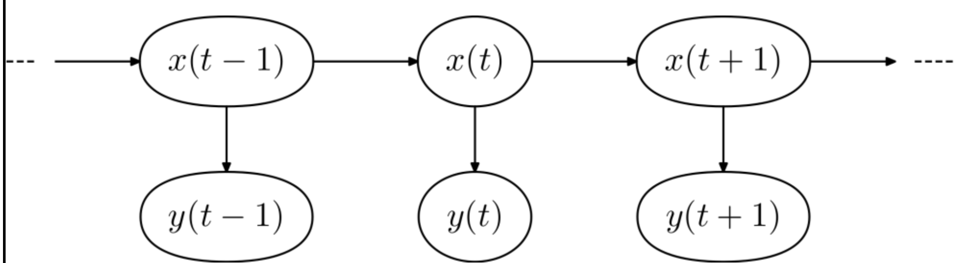
What if we use whole words as symbols?

- This removes the spelling mistakes, and lets the model cobble together partially syntactically correct sentences
- https://blog.codinghorror.com/markov-and-you/

- Would it ever pass the turing test?
  - If we let our standards slip a bit…
- https://filiph.github.io/markov/

The Hidden Markov Model

- What if our outputs are driven by some hidden process states
  - Think of the internal states of a state space model – we may not be able to directly measure them
  - Optical Character recognition – the hidden state is the character, the observed state is an image
- The HMM handles this process



Next Time…

- **Estimation! (Kalman Filters!)**
- **Digital Control!**
- Review:
  - Chapter 12 of Lathi
  - FPE Chapter 1 and 2
- Ponder?

$$y[k] = f[k] * h[k] \qquad\qquad Y(\Omega) = F(\Omega)H(\Omega)$$

where $F(\Omega)$, $Y(\Omega)$, and $H(\Omega)$ are DTFTs of $f[k]$, $y[k]$, and $h[k]$, respectively; that is,

$$f[k] \Longleftrightarrow F(\Omega), \quad y[k] \Longleftrightarrow Y(\Omega), \quad \text{and} \quad h[k] \Longleftrightarrow H(\Omega)$$

Summary

- FT of sampled data is known as
  - discrete-time Fourier transform (DTFT)
  - discrete in time
  - continuous & periodic in frequency
- DFT is sampled version of DTFT
  - discrete in both time and frequency
  - periodic in both time and frequency
    - due to sampling in both time and frequency
- DFT is implemented using the FFT
- Leakage reduced (dynamic range increased)
  - with non-rectangular window functions

Summary

- FFT exploits symmetries in the DFT
  - Successively splits DFT in half
    - odd and even samples
  - Reduction to elementary butterfly operation
    - with 'twiddle factors'
  - Reduce computations from N^2 to $\left(\frac{N}{2}\right)\log_2(N)$ ☺

- FFT can be used to implement DFT for
  - PSD estimates (periodogram and correlogram)
  - Circular (fast) convolution (and correlation)
    - Requires zero padding to obtain "correct" answer