Signal Processing

သတိ: စာကိုမကျက်ပါနှင့်။
နားလည်အောင်ဖတ်ပြီးစဉ်းစားပါ။

Signal and Symbol

- Signal ဆိုတာ Information ပေးနိုင်တဲ့ ဘယ် Physical Entity (Activity) မဆိုကို ခေါ်တာ ဖြစ်ပါသည်။ (Anything physical that carries information: electrical, chemical, gestures...) ။ ဥပမာ၊ ခေါင်းလောင်းတီး၊ ဟွန်းတီး၊ တခါးခေါက်၊ အသံလွှင့်၊ အပူချိန်၊ မိုးရေချိန်။

- Symbol ဆိုတာက Contextual Information ပေးနိုင်တဲ့ Signal တစ်ခု ဖြစ်ပါသည်။ ဥပမာ၊ Smiley ( ) တစ်ခုသည် Symbol တစ်ခု ဖြစ်သည်။ သာမန် Signal နှင့်ကွာသည်က Symbol များသည် Contextual (Context ပေါ်မူတည်သည်) ဖြစ်သည်။ ပို၍ရှင်းအောင် ပြောရင် Symbol ဆိုတာက တင်စားမှုများ (တင်စားထားသော Signal) ဖြစ်သည်။ Text နှင့် Number များကို အတိုင်းအတာ တစ်ခုအထိ Symbol များဟု ယူဆလို့ရသည်။

- ဆိုတော့ Signal သည် Information များကို Encode လုပ်ပြီး Carry လုပ်ပါသည်။ တဖန် Signal များက Carry လုပ်ထားသော Information များကို နားလည်ဖို့ Decode ပြန်လုပ်ရပါသည်။

- Encoded Signal များကို အများအားဖြင့် Data များဟု ခေါ်ပါသည်။ အမှန်တော့ Signal များရော၊ Data များကိုပါ ထပ်ခါထပ်ခါ Encode / Decode လုပ်လို့ရပါသည်။

- ဘာနဲ့တူလဲ ဆိုတော့။ Package များနှင့် တူပါသည်။ Package ထဲသို့ ထည့်ခြင်းသည် Encode လုပ်ခြင်း ဖြစ်ပြီး Package ထဲမှ ထုတ်ခြင်းသည် Decode လုပ်ခြင်း ဖြစ်သည်။ ဆိုတော့ Package သေးသေးလေးကို Package အကြီးကြီးထဲ ထပ်ထည့်လို့ ရသလိုမျိုး ဖြစ်သည်။

Data

- 5.01, 5.02, 5.04, 5.02, 5.34, ...
- ဒီအတိုင်းဆိုရင် ဒီဂဏန်းများသည် ဘာကို ဆိုလိုမှန်း သိမည်မဟုတ်။ ဒါသည် Data ဖြစ်သည်။ Data များသည် Encoded Signal (or Symbol) များ ဖြစ်သည်။
- ဒီ Data များသည် အတန်းထဲရှိ ကျောင်းသား၊ ကျောင်းသူများ အရပ်ကို ကိုယ်စားပြုသည်လို့ ပြောရင် ခင်ဗျားသိသွားပြီ။ Signal များကို Decode လုပ်လိုက်တာ ဖြစ်သည်။
- Data များသည် အများအားဖြင့် Text နှင့် Number များ ဖြစ်ကြသည်။ အကြမ်းအားဖြင့် Encoded Signal များမှာမှ Text နှင့် Number များကိုသာ Data ဟု ခေါ်ကြတာ များပါသည်။
- အမှန်တော့ ဒီ Data များကို ထပ်ပြီး Process လုပ်လို့ ရပါသည်။ Statistics ကိုသုံးပြီး Probability Distribution ရှာတာမျိုး။
- ဆိုတော့ Signal များကို Process လုပ်ခြင်းကို Signal Processing ဟု ခေါ်ပြီး Data များကို Process လုပ်ခြင်းကို Data Processing ဟုခေါ်ပါသည်။

Information

- Signal များက Information များကို Carry လုပ်သည်ဟု ပြောခဲ့ပါသည်။ ဒါဆိုရင် Information ဆိုတာ ဘာကြီးလဲ။

- သတင်း (Information) ဟူသည် မမျှော်လင့်ထားသော၊ ဖြစ်လေ့မရှိသော အကြောင်းအခြင်း အရာများလို့ ဆိုနိုင်ပါသည်။ ဆရာကြီးတစ်ယောက် ပြောဖူးပါသည်။ လူကို ခွေးကကိုက်လျှင် သတင်းမဟုတ်၊ ခွေးကို လူကကိုက်လျှင်သာ သတင်း ဖြစ်သည်။

- ဆိုတော့ Signal များက Encode လုပ်ထားတဲ့ Information များကို Decode လုပ်လိုက်လို့ သိထားပြီးသားကြီး ဆိုရင် Signal များမှာ ပါဝင်သော Information Content သည် နည်းပါလိမ့်မည်။ Signal များက Encode လုပ်ထားတဲ့ Information များကို Decode လုပ်လိုက်လို့ မမျှော်လင့်ထားတာဆိုရင် ပါဝင်သော Information Content သည် များပါလိမ့်မည်။

- ဒီအချက်ပေါ်မူတည်ပြီး Information ကို ကျွန်တော်တို့ Define လုပ်ကြပါသည်။ Information ဆိုတာကို Measurement of Surprise လို့ အကြမ်းဖျဉ်း ယူဆနိုင်ပါသည်။

- **Less Probable** Event တစ်ခုသည် **More Surprising** ဖြစ်သည့်အတွက် Information သည် Random Event များနှင့် ဆိုင်ပါသည်။

- ဥပမာ၊ ခေါင်းပန်းလှန်ခြင်းနှင့် မနက်ဖြန်မိုးရွာ၊ မရွာ ခန့်မှန်းခြင်းကို Yes/No ဖြင့် ဖော်ပြ (Encode)နိုင်သော်လည်း Information Content ခြင်းမတူပါ။

Information Theory

$$H(x) = \sum P(x)[-log_b(P(x))]$$

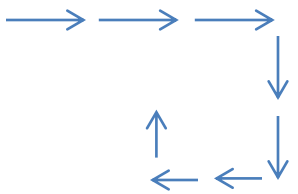
- Claude Shannon က Information Theory ကို အဆိုပြုခဲ့ပါသည်။ Information Theory သည် Entropy အပေါ် အခြေခံပါသည်။
- Information တစ်ခုကို Encode လုပ်ဖို့ လိုအပ်သော အနည်းဆုံး Signal သည် Yes/No (ခေါင်းငြိမ့်၊ ခေါင်းခါ) ဖြစ်သည်။ ဒါကို Bit ဟုခေါ်ပြီး b = {1, 0} (Binary Value) ဖြစ်သည်။
- အမှန်တော့ Encode လုပ်ဖို့ လိုအပ်တဲ့ Information ကိုယ်တိုင်သည် Random Event တစ်ခု ဖြစ်သည်။ ဥပမာ၊ မိုးရွာခြင်း စသည်။ ဒါကြောင့် ဒါကို Probability P(x) ဖြင့် ဖော်ပြနိုင်သည်။
- Probability P(x) ရှိတဲ့ Random Event တစ်ခုကို Bit [b = {1, 0}] ဖြင့် ဖော်ပြမည်ဆိုလျှင် $\log_b P(x)$ (Binary Logarithm of P(x)) အရေအတွက်ရှိတဲ့ Bits လိုသည်။ ဆိုလိုတာက Random Event တစ်ခုကို Binary Value ဘယ်လောက်များများဖြင့် ဖော်ပြဖို့ လိုမည်လဲကို ဆိုလိုသည်။ မိုးရွာခြင်းကို ဖော်ပြဖို့ (ခေါင်းငြိမ့်၊ ခေါင်းခါ) 1 Bit of Encoding လိုသော်လည်း မိန်းကလေးတစ်ယောက်လှပခြင်းကို ဖော်ပြဖို့ More than 1 Bit လိုပါလိမ့်မည်။ $\log_b P(x)$ သည် Number of Bits to represent P(x) ဖြစ်သည်။
- သို့သော် Probability နှင့် Measure of Surprise က Inversely Proportional (Surprise = 1 / Probability) ဖြစ်သည့်အတွက် $\log_b(1/P(x)) = -\log_b P(x)$ ဖြစ်သွားသည်။ ဒါကြောင့် Probability နည်းလျှင် ဖော်ပြဖို့ လိုအပ်တဲ့ Number of Bits များမည် ဖြစ်သည်။
- ထို့ကြောင့် Information Content (Measurement of Surprise) သည် Random Event တစ်ခု၏ Probability Distribution နှင့် ထို Probability ကို ဖော်ပြဖို့ လိုအပ်တဲ့ (Negative) Number of Bits တို့ မြောက်လဒ်ဖြစ်သည်။ ဒါကို Information Entropy ဟုခေါ်သည်။ ဒါသည်ပင် Information Content ဖြစ်သည်။

Example

- N, E, W, S ဆိုပြီး Symbols 4 ခုရှိသည့်အတွက် တစ်ခုခြင်းအတွက် (Uniform) Probability P(x) သည် ¼ ဖြစ်သည်လို့ ယူဆလိုက်သည့်အခါ $\log_b(1/¼) = \log_b(4)$ သည် 2 ဖြစ်သည်။ ထို့ကြောင့် N, E, W, S ကို 2 Bit ဖြင့် Encode လုပ်လို့ရသည်။
- ဟုတ်ပြီ။ Traversal T = {E, E, E, S, S, W, W, N }ဆိုပါတော့။ ဒါဆိုရင် 01 01 01 11 11 10 10 00 ဆိုပြီး ဖြစ်သွားမည်။
- ထို့ကြောင့် P(x) သည်အမြဲ ¼ ဖြစ်သည်လို့ ယူဆပြီး Information Entropy ကိုရှာကြည့်လျှင်လည်း H(x) = ¼ (2) + ¼(2) + ¼(2) + ¼(2) = 2 Bit ဖြစ်သည်။
- သို့သော် Traversal ရဲ့ တကယ့် Probability Distribution က P(x) = {1/8, 3/8, 2/8, 2/8 } ဖြစ်သည်။
- ဒီတခါ တကယ့် Probability Distribution ကိုကြည့်ပြီး Information Entropy ကိုရှာကြည့်လျှင် $H(x) = 1/8(\log_b(8/1)) + \ldots 2/8(\log_b(8/2)) = 1.905$ Bit ရမည်ဖြစ်သည်။
- ဘာဖြစ်လို့လဲဆိုတော့ P(E) သည် P(N) ထက်ပိုကြီးလာတဲ့အခါ သူ့ရဲ့ Measure of Surprise ကလျော့သွား၍ ဖြစ်သည်။
- ထို့ကြောင့် Uniform Probability Distribution သည် Information Content အများဆုံး ဖြစ်သည်။ Zero Probability Distribution သည် Information Content အနည်းဆုံး (Zero) ဖြစ်သည်။

Actual	Symbol	Bits
↑	N	00
→	E	01
←	W	10
↓	S	11

Traversal



Traversal Distribution

N	E	W	S	Total
1	3	2	2	8

Cross Entropy

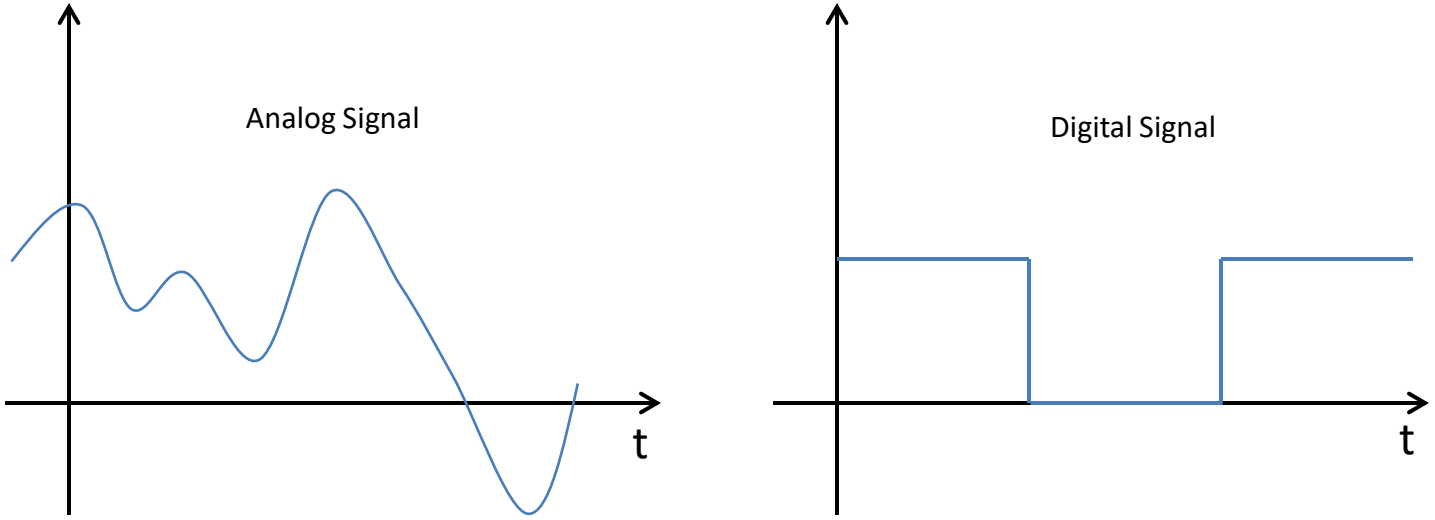
$$H(p,q) = \sum P(x)[-log_b(Q(x))]$$

- တကယ်လို့သော Probability Distribution P(x) နှင့် The Probability Q(x) မတူဘူးဆိုရင် ဒါကို Cross Entropy ဟုခေါ်သည်။
- ပို့ရှင်းအောင် ပြောလျှင် စာတွေ့နှင့် လက်တွေ့ သဘောမျိုး ဖြစ်သည်။ Q(x) က ကနဦး သိရှိထားသော (Learned) လုပ်ထားသော Probability Distribution ဖြစ်ပြီး ဒါကို Number of Bits များဖြင့် ဖော်ပြပါသည်။ P(x) က Real (လက်တွေ့) Probability Distribution ဖြစ်သည်။ စာတွေ့ကသိသော Information နှင့် လက်တွေ့ကသိသော Information တို့မတူ။ Cross Entropy က ထိုသဘောတရားကို ဆိုလိုသည်။
- ဒါကိုနောက်ပိုင်း Learning လုပ်တဲ့အခါ ဖော်ပြကြပါသည်။ ဒီမှာတော့ Introduction လောက်ပဲ ပြောပါမည်။ နောက်မှ အသေးစိတ် ပြောပါမည်။
- အခု သိဖို့လိုအပ်သည်က Information Entropy သည် Signal များမှာ ပါဝင်တဲ့ Information Content ကိုဆိုလိုကြောင်းကို သိရင်ရပါသည်။

Signal Processing

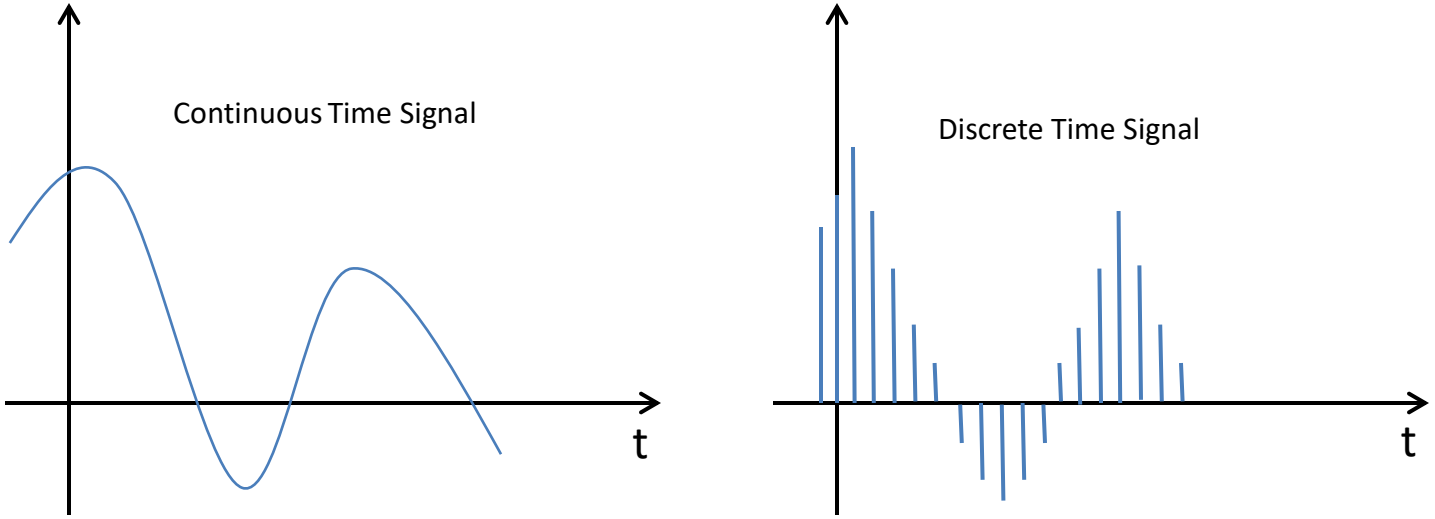
- အခု ကျွန်တော်တို့ Signal၊ Data နှင့် Information တို့အကြောင်းကို နဲနဲလောက် သိသွားပါပြီ။
- ကျွန်တော်တို့ ဆက်ပြီးတော့ Signal များကို ဘယ်လို Process လုပ်မလဲကို လေ့လာမည် ဖြစ်သည်။
- ဒါကိုပင် Signal Processing ဟုခေါ်ပါသည်။ Signal Processing သည် ICT (Information and Communication Technologies) အားလုံးရဲ့ Foundation ဖြစ်သည်လို့ ပြောရင်မမှားပါဘူး။
- Signal Processing ကြောင့်သာ ကျွန်တော်တို့ Mobile Phone တွေ၊ Internet တွေ၊ Computer တွေသုံးလို့ ရတာ ဖြစ်သည်။
- ထို့အပြင် ကျွန်တော်တို့ Image Processing၊ Speech Processing တို့ လုပ်ဆောင်မည် ဆိုလျှင်လည်း Image Signal များ၊ Speech Signal များကိုပင် Process လုပ်ရမည်။ ထို့ကြောင့် ဒါသည် အင်မတန် အရေးကြီးပါသည်။
- သို့သော် Signal Processing အကြောင်းပြောလျှင် Signal အကြောင်းချည်းပဲပြောလို့မရတော့။ Signal များကို Process လုပ်မည့် Systems များအကြောင်းကိုပါ ပြောရမည် ဖြစ်သည်။ ထို့ကြောင့် Signal နှင့် Systems အကြောင်းကို ဆက်ပြီး ဆွေးနွေးပါမည်။

Analog and Digital Signals



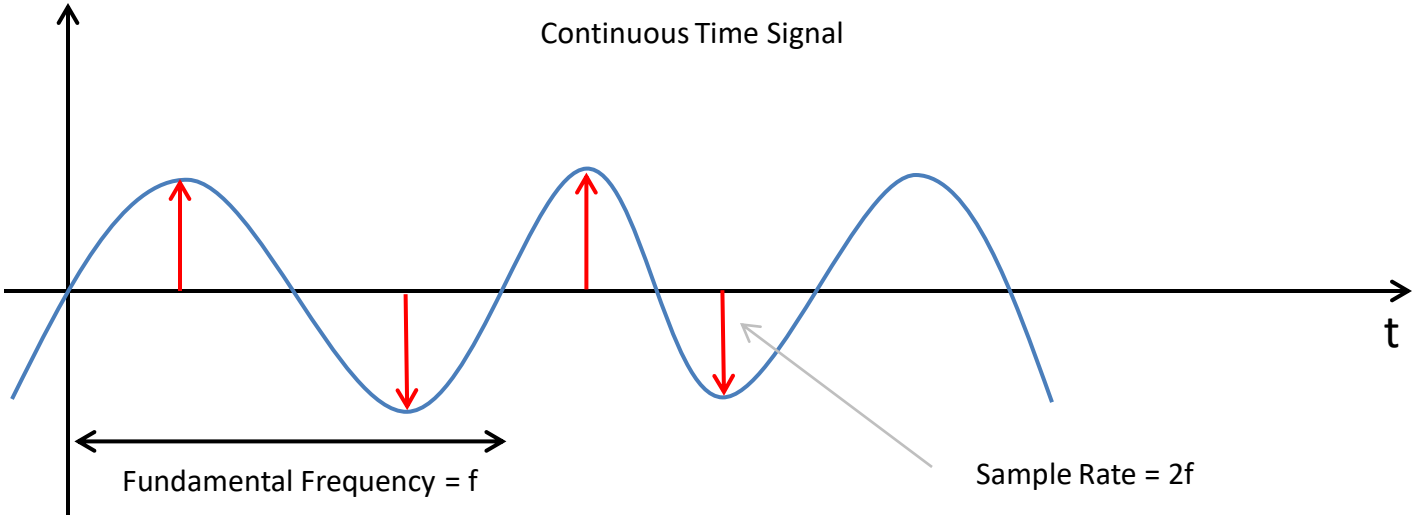
- Signal တွေကို သူတို့ရဲ **State** ပေါ်မူတည်ပြီး Digital Signals များနှင့် Analog Signals များဆိုပြီး ခွဲနိုင်ပါသည်။
- Signal တစ်ခုက Binary States ပဲရှိတယ်ဆိုရင် ဒါကို Digital Signals များလို့ ခေါ်ပါသည်။
- Signal တစ်ခုက Multiple States (More than Binary) ရှိတယ်ဆိုရင်တော့ ဒါကို Analog Signals များလို့ ခေါ်ပါသည်။

Continuous and Discrete Time Signals



- Signal တွေကို သူတို့ရဲ့ **Time** ပေါ်မူတည်ပြီး Continuous Time Signals များနှင့် Discrete Time Signals များဆိုပြီး ခွဲနိုင်ပါသည်။
- Signal တစ်ခုသည် Continuously အချိန်နှင့် အမျှပြောင်းလဲနေလျှင် Continuous Time Signal များဖြစ်ပါသည်။
- Signal တစ်ခုသည် Time Intervalနှင့်သာ ပြောင်းလဲနေလျှင်တော့ Discrete Time Signal များဖြစ်ပါသည်။ အမှန်တော့ Discrete Time Signal များကို Time Intervalအလိုက် Sample ယူထားခြင်းဖြစ်သည်။ ဥပမာ၊ နေ့အပူချိန်များသည် နေ့အလိုက် အပူချိန်များသာ ဖြစ်ပြီး အချိန်နှင့် အမျှပြောင်းလဲနေသော အပူချိန်များ မဟုတ်ပါ။

Nyquist-Shannon Sampling Theorem



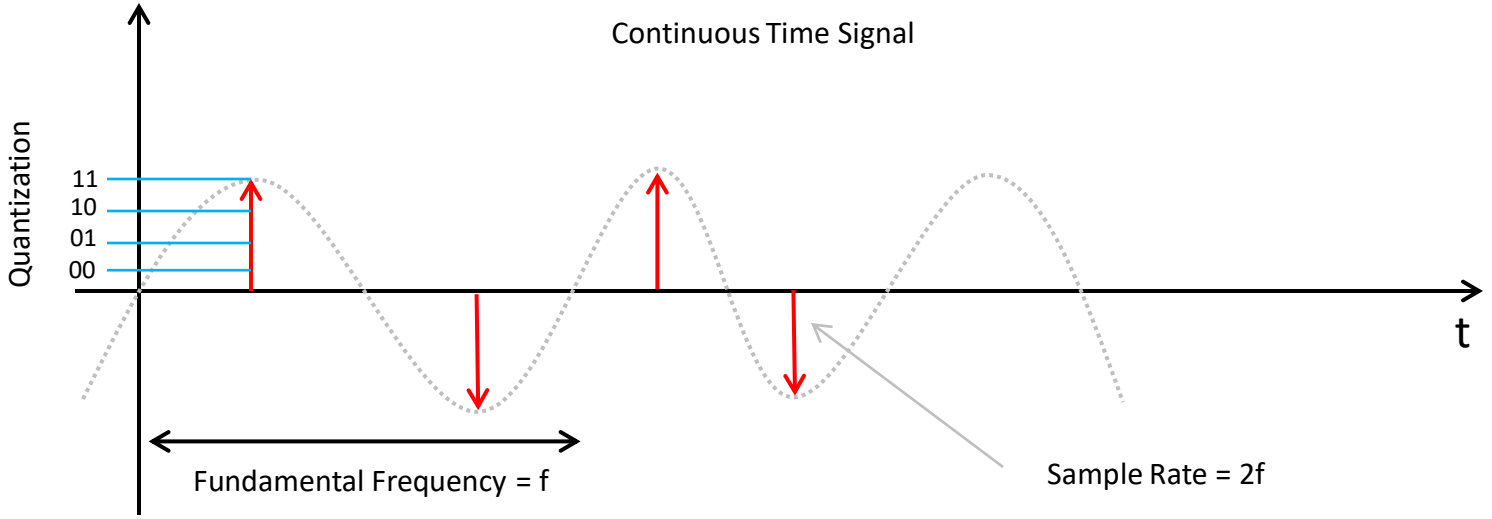
- Continuous Time Signal များကို Discrete Time Signal သို့ ပြောင်းမည်ဆိုရင် Nyquist-Shannon Sampling Theorem ကို သုံးရပါသည်။

- Continuous Time Signal တစ်ခုရဲ့ Fundamental Frequency တစ်ခုသည် f (Hz) ဆိုပါတော့၊ ဒါဆိုရင် Continuous Time Signalကို Discrete Time Signal ပြောင်းဖို့လိုအပ်တဲ့ Sample Rate သည် Fundamental Frequency၏ 2 ဆ (2f) ဖြစ်ရမည်။ Fundamental Frequency ကို ကျွန်တော်တို့ နောက်မှ ပြောပါမည်။

- ဥပမာ၊ နေ့တစ်နေ့၏ အပူချိန်များကို သိလိုလျှင် အချိန်တိုင်း အပူချိန်တိုင်းစရာမလို။ တနေ့မှာ အပူဆုံးချိန်နှင့် အအေးဆုံးအချိန် အပူချိန်များကို တိုင်းလျှင်ရပြီ ဖြစ်သည်။ နေ့တစ်နေ့ Fundamental Frequency ဖြစ်ပြီး တနေ့ 2 ကြိမ်တိုင်းခြင်းသည် Sampling Rate ဖြစ်သည်။ တနေ့၏ အပူဆုံးအချိန်နှင့် အအေးဆုံးအချိန်ကို သိလျှင် တနေ့လုံးအပူချိန်ကို ပြန်ခန့်မှန်းကြည့် (Reconstruct) လို့ ရသည်။

- Sampling Rate သည် Fundamental Frequency ထက်နည်းလျှင် Under Sampling ဖြစ်ပြီး Sampling Rate သည် Fundamental Frequency ထက်များလျှင် Over Sampling ဖြစ်မည်။ ထိုအခါ Signal Reconstruction လုပ်သည့်အခါ Aliasing များဖြစ်လာမည်။

Pulse Code Modulation



- Analog (Multi State) Signal များကို Digital Signal (Binary State) Signal များအဖြစ်ပြောင်းလိုလျှင် ပထမ Analog Signal များကို Discrete Time Signal များအဖြစ်ပြောင်းရပါသည်။

- ရရှိလာသော Discrete Time Signal များကို Quantization လုပ်ရပါသည်။ Quantization လုပ်ခြင်းသည် ရရှိလာသော Discrete Time Signal များ၏ Amplitude ကို Binary Code (Bit) များအနေဖြင့် ဖော်ပြခြင်းသာ ဖြစ်သည်။

- ထိုသို့ Analog (Multi State) Signal များကို Digital Signal (Binary State) Signal များအဖြစ်ပြောင်းခြင်းကို Pulse Code Modulation ဟုခေါ်သည်။

- ဤနည်းဖြင့် ကျွန်တော်တို့ Signal များကို Digitization လုပ်ယူတာ ဖြစ်ပါသည်။ ဒါကြောင့် အခုခေတ်ကြီးကို Digital ခေတ်လို့ ခေါ်တာဖြစ်ပါသည်။

Signal Analysis

Time \ State	Analog (Multi)	Digital (Binary)
Continuous	Yes	Yes
Discrete	Yes	Yes

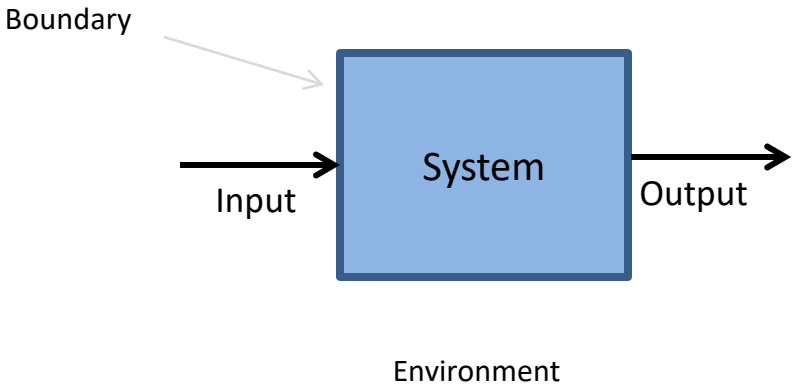
- Signal များ လေ့လာရာ (Analysis) မှာ Domain 3 ခုရှိပါသည်။ Space Domain, Time Domain နှင့် Frequency Domain ဆိုပြီးတော့ ဖြစ်သည်။

- ဥပမာ၊ ကျွန်တော်တို့ စကားပြောသည် ဆိုလျှင် အမှန်တော့ Sound Wave Signal များကို ထုတ်လွှင့်လိုက်ခြင်းသာ ဖြစ်သည်။ Sound Wave များသည် ကျွန်တော်ပြောတဲ့ နေရာမှ အခြားနေရာသို့ ပျံ့သွား (Propagate) ပါသည်။ Wave Propagation ကို လေ့လာလျှင် Space Domain ဖြစ်သည်။ ပြီးရင် ကျွန်တော့်စကားသံ (အသံအနိမ့်အမြင့်) သည် အချိန်နှင့်အမျှလဲ ပြောင်းနေသည်။ ဒါသည် Time Domain ဖြစ်သည်။ ကျွန်တော်သည် ယောကျ်ားလေး ဖြစ်သည့်အတွက် မိန်းကလေး အသံနှင့် တူမည်မဟုတ်။ထိုသို့ Sound Wave များ၏ Pitch (Frequency) များကို လေ့လာခြင်းသည် Frequency Domain ဖြစ်သည်။

- သို့သော် လက်တွေ့ Signal Analysis မှာတော့ ကျွန်တော်တို့ **Time Domain** နှင့် **Frequency Domain** ကိုသာလေ့လာကြပါသည်။

- Signal အကြောင်းများကို နည်းနည်းသိသွားပြီ ဖြစ်သည့်အတွက် Signal များကို Process လုပ်မည့် Systems များကို ဆက်ပြီးဆွေးနွေးကြပါစို့။

System

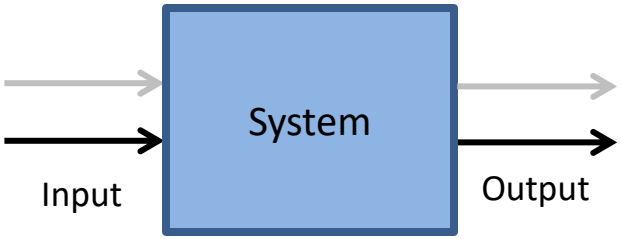


- System သည် Components များပါဝင်သော အရာတစ်ခုဖြစ်ပြီး သူ၏ Environment ကို Boundary နှင့် ပိုင်းခြားထားပါသည်။ System တစ်ခုက သူ၏ Environmentမှ Input များကို Process လုပ်ပြီး Output များကို ထုတ်ပေးပါသည်။

- သန့်ရှင်းနည်းဖြင့် System တစ်ခုကို အလွယ်ဆုံး ဖော်ပြမည်ဆိုပါက Mathematical Function တစ်ခုသာ ဖြစ်ပါသည်။

$$y = f(x)$$

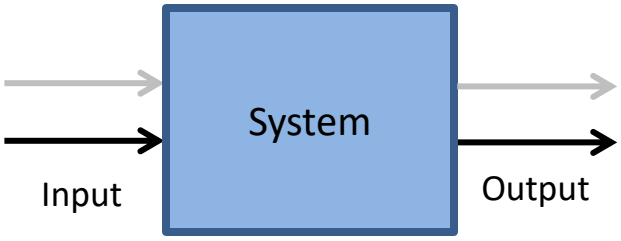
System Inputs and Outputs



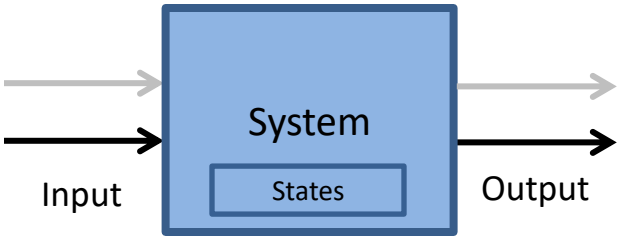
Output \ Input	Single	Multiple
Single	Yes	Yes
Multiple	Yes	Yes

- System တစ်ခု၏ Input နှင့် Output များသည် Single သို့မဟုတ် Multiple ဖြစ်နိုင်ပါသည်။
- System တစ်ခုက Multiple Output ဆိုပါက Vector Function တစ်ခုပြီး Single Output ဆိုပါက Scalar Function တစ်ခုဖြစ်သည်။ (Advanced Calculus ကို ပြန်ကြည့်ပါ။)

Static and Dynamic Systems



Static Systems

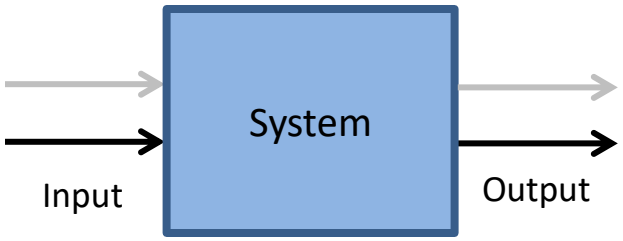


Dynamic Systems

- System တစ်ခုမှာ Internal States များမရှိဘဲ Input များကို တိုက်ရိုက် Process လုပ်ပြီး Output ထုတ်ရင် ဒါကို Static Systems ဟုခေါ်ပါသည်။
- System တစ်ခုမှာ Internal States များပါဝင်ပြီး Output များသည် Input ပေါ်မှာရော Internal States (Previous States) များပေါ်မှာပါ မူတည်ရင် ဒါကို Dynamic Systems ဟုခေါ်ပါသည်။
- Dynamic Systems များကို Calculus ဖြင့် ဖော်ပြရပါသည်။ (Calculus ကိုပြန်ကြည့်ပါ။)

Output = Previous **State** + **Input**

Linear and Non Linear Systems

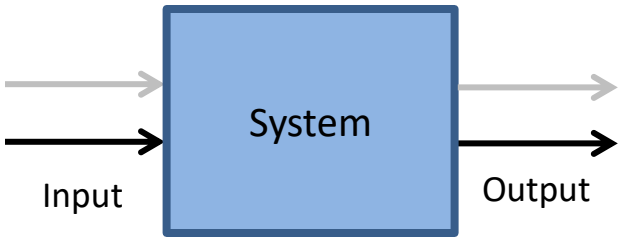


- System တစ်ခုသည် Input များကို Output များသို့ Linearly Map (Transform) လုပ်ပေးနိုင်ပါက ဒီ System ကို Linear System ဖြစ်သည်ဟု ခေါ်သည်။
- ထို့ကြောင့် Linear System တစ်ခုကို Linear Transformation ဖြင့်ဖော်ပြနိုင်သည်။

$$\mathbf{y} = A\mathbf{x} + \mathbf{b}$$

- System တစ်ခုကို Linear Transformation ဖြင့် မဖော်ပြနိုင်ပါက ထို System ကို Non Linear System ဟုခေါ်သည်။

Time Invariant and Time Varying Systems

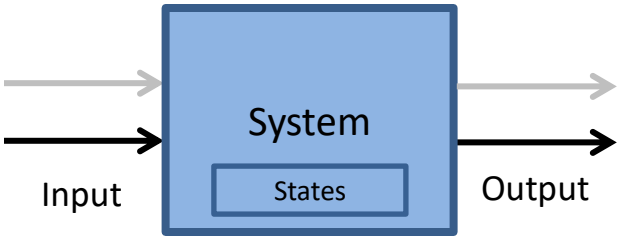


- System တစ်ခုစီ၏ Parameters များသည် အချိန်နှင့်အမျှ မပြောင်းလဲပါက ဒါကို Time Invariant Systems ဟုခေါ်သည်။
- Time Invariant System များသည် Time Delay ဖြစ်သော်လည်း Output များမပြောင်းပါ။

$$f(x) = f(x - t) \text{ if and only if } x = (x - t)$$

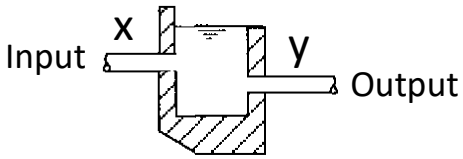
- System တစ်ခုသည် Time Invariant မဖြစ်ပါက Time Varying System များဖြစ်သည်။
- လက်တွေ့မှာတော့ Time Invariant System များသည် မဖြစ်နိုင်ပါ။ မီးသီးများသည် အချိန်ကြာလာလျှင် ကျွမ်းသွားမည်။ Computer များလည်း သုံးပါများလျှင် ပိုနှေးလာမည်။ သို့သော်လည်း တွက်သည့်အခါ လွယ်ကူအောင် Time Invariant System များကိုသာ စဉ်းစားလေ့ရှိပါသည်။

Linear Time Invariant Dynamical Systems



- Linear Dynamical System ဆိုသည့်အတိုင်း: Linear ရော၊ Dynamic ရော၊ Time Invariant ရော ဖြစ်တဲ့ System များကို ဆိုလိုပါသည်။
- အမှန်တော့ ကျွန်တော်တို့ လေ့လာမည်က System များသည် Linear Time Invariant Dynamical System များဖြစ်ပါသည်။
- ဒါဆိုရင် Linear Time Invariant Dynamical System များကို ဘယ်လို ဖော်ပြမလဲ။

Simple Differential Equations



- ဒီရေတိုင် တစ်ခုထဲကို x(t) နှုန်းဖြင့် ရေသွင်းမည် ဆိုပါတော့။ ရေတိုင်ကီထဲမှ ရေမထုတ်ဘူးဆိုပါက ရေတိုင်ကီ ပြည့်လာမည် ဖြစ်သည်။ ထိုပြည့်လာသော ရေသည်

$$\mathbf{v} = \int x(t)dt$$

- နောက်တစ်ခါ ရေမသွင်းဘဲ ရေတိုင်ကီထဲမှ y(t) နှုန်းဖြင့် ရေထုတ်မည်ဆိုပါက ထွက်သွားမည့်ရေသည်

$$\mathbf{u} = \int y(t)dt$$

- သို့သော် y(t)သည် ရေဖိအားပေါ်မူတည်သည့်အတွက် ရေအမြင့်ပေါ်မူတည်သည်။ ရေထွက်နှုန်းသည် ရေအမြင့် h(t) နှင့် Directly Proportional ဖြစ်သည့်အတွက်

$$\mathbf{y(t) = kh(t) \Rightarrow h(t) = cy(t)} \text{ , where } c = 1/k, \; k = \text{constant}$$

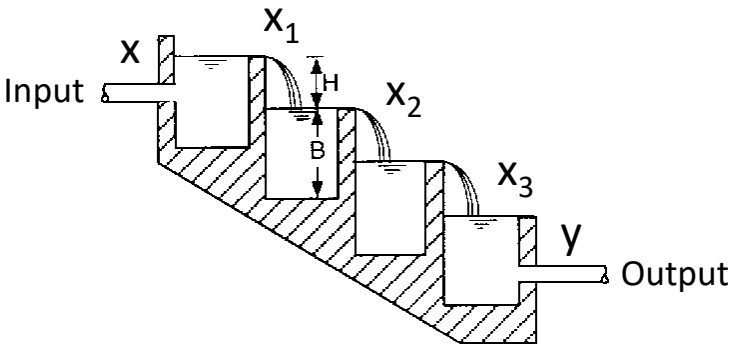
- x(t) နှုန်းဖြင့် ရေသွင်းပြီး y(t) နှုန်းဖြင့် ရေထုတ်မည်ဆိုပါက ရေတိုင်ကီထဲရေသည်

$$\int x(t)dt - \int y(t)dt = \mathbf{Ah(t)} \text{ where A = base area = constant}$$

$$\int x(t)dt - \int y(t)dt = \mathbf{ay(t)} \text{ where a = Ac = constant}$$

$$\mathbf{x(t) = y(t) + a}\frac{d}{dt}y(t)$$

Linear Differential Equations



- ကျွန်တော်တို့ ရေတိုင်ကီ ၄ လုံးကို ဒီလိုဆင့်ထားမယ်ဆိုရင် နောက်ဆုံးရေတိုင်ကီ၏ Input သည် တတိယရေတိုင်ကီ၏ Output ဖြစ်သည်။

$$x_3(t) = y(t) + a\frac{dy}{dt}$$

$$x_2(t) = x_3(t) + b\frac{dx_3}{dt}$$

$$x_2(t) = y(t) + (a + b)\frac{dy}{dt} + ab\frac{d^2y}{dt^2}$$

- ဒါကြောင့် x(t) သည် ဒီလို Form ရှိပါသည်။ ဒါကို Linear Differential Equation ဟုခေါ်ပါသည်။

$$x(t) = y(t) + a_0\frac{dy}{dt} + a_1\frac{d^2y}{dt^2} + a_2\frac{d^3y}{dt^3} + a_3\frac{d^4y}{dt^4}$$

General Linear Differential Equations

$$x(t) = a_0 y(t) + a_1 \frac{dy}{dt} + a_2 \frac{d^2y}{dt^2} + a_3 \frac{d^3y}{dt^3} + \dots + a_n \frac{d^ny}{dt^n}$$

- ကျွန်တော်တို့ Linear Differential Equations များသည် Linear Dynamical System များကို ဖော်ပြနိုင်ကြောင်း သိရှိလာပြီ ဖြစ်သည်။
- a_0, a_1, a_2, …, a_n တို့သည် Parameter များဖြစ်ပြီး Parameter များ၏ တန်ဖိုးသည် အချိန်နှင့် အမျှမပြောင်းလဲပါက ဒါကို Time Invariant ဖြစ်သည်လို့ ဆိုပါသည်။
- Linear Differential Equations များတွင် a_0, a_1, a_2, …, a_n တို့က အမြဲ Constant ဖြစ်သည့်အတွက် Time Invariant ဖြစ်သည်လို့ ဆိုနိုင်ပါသည်။
- Linear Differential Equations များဖြင့် Linear Time Invariant Dynamical System များကို ဖော်ပြနိုင်ပါသည်။
- The Order of Differentiation သည် Previous States ဘယ်နှခုကို မှီခိုလဲကို ကိုယ်စားပြုပါသည်။ ရေတိုင်ကီ 4 လုံးတွင် Previous States 4 ခုပေါ်မူတည်သဖြင့် 4^{th} Order of Differentiation လိုပါသည်။ သို့သော် ရေတိုင်ကီ 1 လုံးတွင် Previous States 1 ခုပေါ်သာ မူတည်သဖြင့် 1^{st} Order of Differentiation လိုပါသည်။

Linear Differential Operator

$$x(t) = y(t)(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n)$$

- ကျွန်တော်တို့ Linear Algebra မှာ Differential များသည် Vector Space မှာပါဝင်ပြီး Linear Mapping (Linear Operator) အဖြစ်ဖော်ပြလို့ ရကြောင်း ပြောခဲ့ပါသည်။

- $(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n)$ ကို Linear Mapping (Matrix) တစ်ခု အဖြစ်ဖော်ပြပါက

$$\mathbf{x} = M\mathbf{y}$$

- ဒါသည်ပင် Linear Time Invariant Dynamical Systems များကို ဖော်ပြနိုင်တဲ့ အခြေခံ Linear Mapping တစ်ခု ဖြစ်ပါသည်။

Linear Dynamical Component

$$y(t) = a_0 x(t) + a_1 \frac{dy}{dt}$$

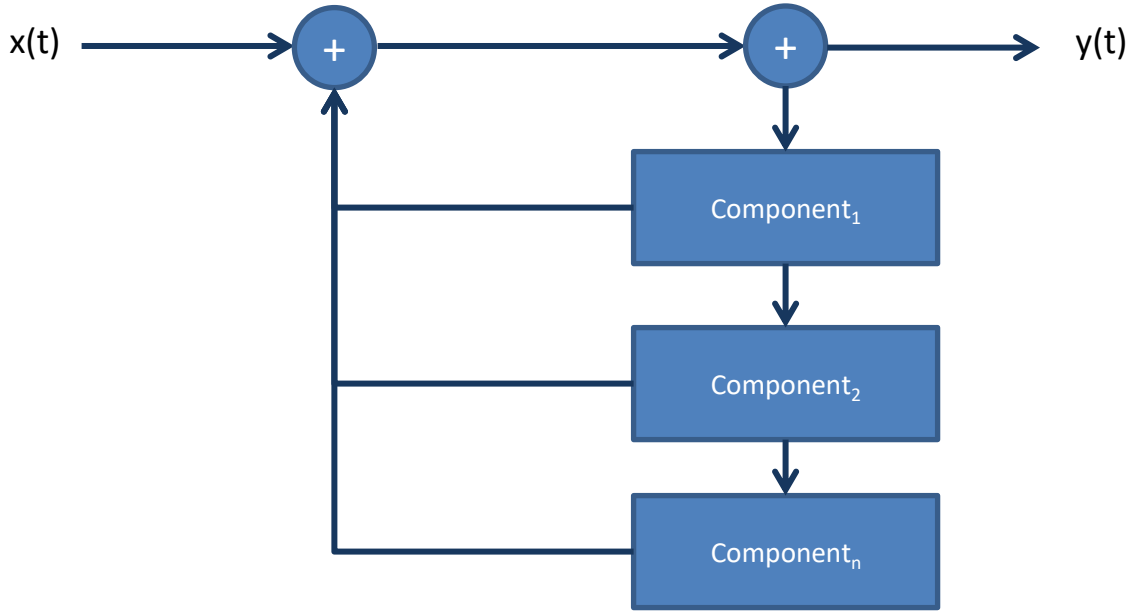
$$y[n] = a_0 x[n] + a_1 y[n-1]$$

- Linear Dynamical Component တစ်ခုသည် အောက်ပါ Properties များရှိပါသည်။ Linear Dynamical Component တစ်ခု၏ Output သည် Current Input နှင့် Previous Output တို့၏ Linear Combination ဖြစ်ပါသည်။
- ထိုအချက်ကို လိုက်နာသော ဘယ် Component မဆို Linear Dynamical Component ဖြစ်ပါသည်။ ဥပမာ၊ ပိုက်ဆံအတွက် ဆိုရင် စုဘူး၊ ရေအတွက်ဆိုလျှင် ရေတိုင်ကီ၊ Electric Current များအတွက် ဆိုလျှင် Capacitor။ ဒါတွေအားလုံးသည် Linear Dynamical Component များဖြစ်ပါသည်။
- Linear Dynamical Component များရှိသလို Non-Linear Dynamical Component များလည်းရှိပါသည်။ ဥပမာ၊ အင်မတန် နာမည်ကြီးနေသော Neural Network မှ Neuron တစ်ခုသည် Non-Linear Dynamical Component တစ်ခု ဖြစ်ပါသည်။ ဒါ့အပြင် Neuron တစ်ခုသည် Time Invariant လည်း မဟုတ်ပါ။ ဒါကြောင့် Neuron တစ်ခုသည် Non-Linear Time Variant Component တစ်ခု ဖြစ်ပါသည်။
- Linear Dynamical Component များကို Linear Combination လုပ်ခြင်းအားဖြင့် ကျွန်တော်တို့ Linear Dynamical Network Model ရမှာ ဖြစ်ပါသည်။ ရရှိလာသော Linear Dynamical Network Model ဖြင့် ကျွန်တော်တို့ Linear Time Invariant Dynamical System များကို ဖော်ပြခြင်း ဖြစ်ပါသည်။

Simple LTI System Model (Output Dependent)

$$x(t) = y(t)(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n)$$

$$x[n] = a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + \dots + a_n y[n-m]$$

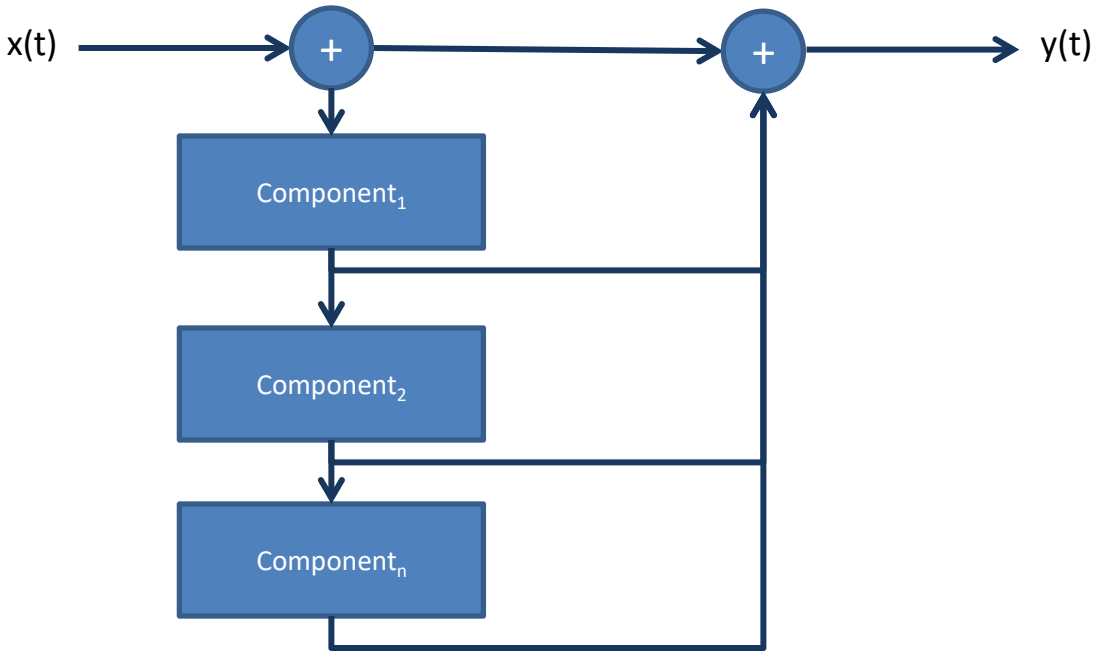


- ကျွန်တော်တို့ Simple LTI System Model ကို Linear Combination of Linear Dynamical Component များဖြင့် ဖော်ပြလို့ ရပါသည်။ ဒါက Output Dependent Model ဖြစ်သည်။

Simple LTI System Model (Input Dependent)

$$y(t) = x(t)(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n)$$

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + \dots + a_n x[n-m]$$

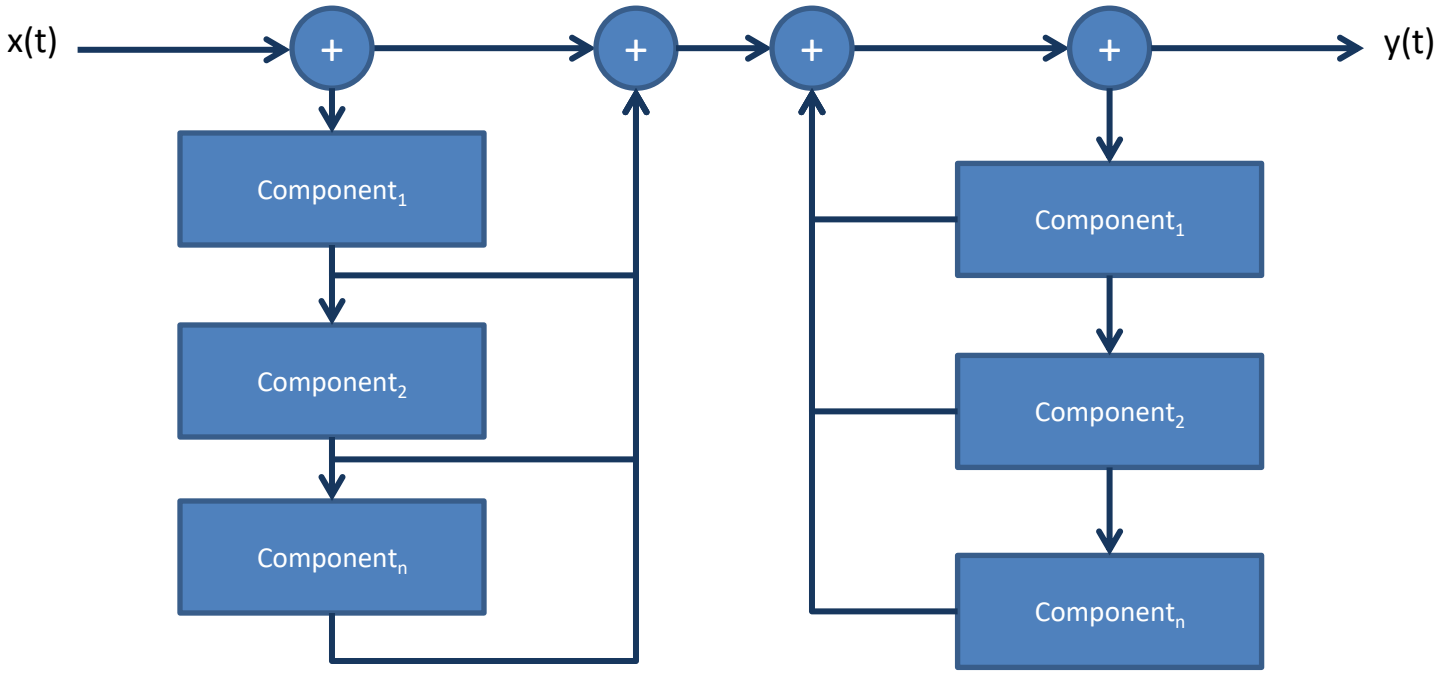


- ကျွန်တော်တို့ Simple LTI System Model ကို Linear Combination of Linear Dynamical Component များဖြင့် ဖော်ပြလို့ ရပါသည်။ ဒါက Input Dependent Model ဖြစ်သည်။

Simple LTI System Model

$$(b_0 D^0 + b_1 D^1 + b_2 D^2 + b_3 D^3 + \ldots + b_n D^n) y(t) = x(t) (a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \ldots + a_n D^n)$$

$$b_0 y[n] + b_1 y[n-1] + b_2 y[n-2] + \ldots + b_n y[n-m] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + \ldots + a_n x[n-m]$$



- ကျွန်တော်တို့ Simple LTI System Model ကို Linear Combination of Linear Dynamical Component များဖြင့် ဖော်ပြလို့ ရပါသည်။ ဒါက Combination of Previous Models များဖြစ်ပြီး Input Output Dependent ဖြစ်ပါသည်။ Simple Model လို့ ဘာလို့ ပြောရတာလဲဆိုတော့ Single Input / Output ဖြစ်လို့ ဖြစ်သည်။

State Space Representation

$$(b_0 D^0 + b_1 D^1 + b_2 D^2 + b_3 D^3 + \dots + b_n D^n) y(t) = x(t) (a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n)$$

$$b_0 y[n] + b_1 y[n-1] + b_2 y[n-2] + \dots + b_n y[n-m] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + \dots + a_n x[n-m]$$

- Linear Dynamical Systems များကို Linear Differential (Difference) Equations များဖြင့် ဖော်ပြနိုင်ကြောင်း ပြောခဲ့ပါသည်။
- အမှန်တော့ Nth Order Linear Dynamical System တစ်ခုသည် Nth Previous State များရှိမည်ဖြစ်သည်။
- တကယ်လို့သော Linear Dynamical System တစ်ခု Input Output Differential Equation ဖြင့် မဟုတ်ဘဲ System ၏ State များဖြင့် ဖော်ပြခြင်းကို State Space Representation ဟုခေါ်ပါသည်။
- ပထမအနေဖြင့် State Space Representation ဖြင့် ဖော်ပြဖို့ အတွက် Nth Order Differential Equation ကို Systems of 1st Order Differential Equation များအဖြစ်ပြောင်းပါသည်။
- ဒုတိယအနေဖြင့် ရရှိလာသော Systems of 1st Order Differential Equation များကို Vector Equation ဖြင့်ဖော်ပြပါသည်။
- ကျွန်တော်တို့က Vector Equation 2 ခုရပါမည်။

$$\mathbf{q}' = A\mathbf{q} + B\mathbf{u}$$

$$\mathbf{y} = C\mathbf{q} + D\mathbf{u}$$

State Equations

$$(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n) y(t) = x(t) (b_0 D^0 + b_1 D^1 + b_2 D^2 + b_3 D^3 + \dots + b_n D^n)$$

$$\begin{aligned} q_1 &= D^0 y \\ q_2 &= D^1 y \\ q_3 &= D^2 y \\ &\dots \\ q_n &= D^{n-1} y \end{aligned}$$

$$\begin{aligned} D^1 q_1 &= q_2 \\ D^1 q_2 &= q_3 \\ D^1 q_3 &= q_4 \\ &\dots \\ D^1 q_n &= -a_{n-1} q_n - a_{n-2} q_{n-1} - \dots - a_0 q_1 + b_0 x + \dots \end{aligned}$$

$$\frac{d}{dt}\begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix}\begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} + \begin{pmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nm} \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}$$

In our Case

$$\begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

- State Equation မှာ Input Vector များက State များကို ဘယ်လိုသက်ရောက်မှုရှိသည်ကို ဖော်ပြပါသည်။
- Vector Equation အရ Derivative of State များကို State Vector (**q**) များနှင့် Input Vector (**u**) တို့ဖြင့်ဖော်ပြနိုင်သည်။

Output Equations

$$(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n) y(t) = x(t) (b_0 D^0 + b_1 D^1 + b_2 D^2 + b_3 D^3 + \dots + b_n D^n)$$

$$q_1 = y$$

$$\begin{aligned} D^1 q_1 &= q_2 \\ D^1 q_2 &= q_3 \\ D^1 q_3 &= q_4 \\ &\dots \\ D^1 q_n &= -a_{n-1} q_n - a_{n-2} q_{n-1} - \dots - a_0 q_1 + b_0 x + \dots \end{aligned}$$

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nm} \end{pmatrix} \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} + \begin{pmatrix} d_{11} & \cdots & d_{1m} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nm} \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}$$

In our Case

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} y \\ \vdots \\ 0 \end{pmatrix} \quad \begin{pmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nm} \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \quad \begin{pmatrix} d_{11} & \cdots & d_{1m} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nm} \end{pmatrix} = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

- Output Equation မှာ Input Vector (**u**) နှင့် State Vector (**q**) များပေါ်မူတည်ပြီး Output Vector ဘယ်လို ရရှိလာသည်ကို ပြောပါသည်။

System Analysis

- Signal များကို လေ့လာသလို System များကိုလည်း Time Domain နှင့် Frequency Domain များပေါ်အခြေခံပြီး လေ့လာကြပါသည်။
- အခုထိ Systems များကို ဖော်ပြသော Linear Differential Equation များနှင့် State Space Representation များသည် Time Domain အပေါ် အခြေခံပါသည်။
- Frequency Domain အကြောင်းကိုတော့ ကျွန်တော်တို့ နောက်မှ ပြောပါမည်။
- အခုလောလောဆယ် System များကို Time Domain များ ဘယ်လို Analysis လုပ်သည်ကို ဆွေးနွေးပါမည်။
- System များကို လေ့လာသည် ဆိုရာတွင် ကျွန်တော်တို့က System Response များကို လေ့လာခြင်းဖြစ်သည်။ ဥပမာ၊ ကျွန်တော်တို့ သစ်သားတုံးကို လက်ဖြင့်ခေါက်ကြည့်လျှင် အသံတစ်မျိုးထွက်လာမည်။ လက်ဖြင့် ခေါက်ခြင်းသည် Input ဖြစ်ပြီး အသံကြားရခြင်းသည် Output ဖြစ်သည်။ တဖန် ကြွေပန်းကန်ကို လက်ဖြင့်ခေါက်ကြည့်လျှင် အသံတစ်မျိုးထပ် ထွက်လာမည်။ သစ်သားတုံးအသံ နှင့် ကြွေပန်းကန်အသံတို့ တူမည် မဟုတ်ပါ။ ဒါဆိုရင် ကျွန်တော်တို့ နားလည်နိုင်သည်က Input တူနိုင်သော်လည်း Output (Reponses) မတူပါ။ System များကွဲပြားခြင်း ဖြစ်ပါသည်။ ဒါကြောင့် System များကို လေ့လာရာတွင် System Response များကို လေ့လာရပါသည်။
- System Response သည် အောက်ပါအချက် တို့ကို ဖော်ပြနိုင်ပါသည်။
  - The Nature of a System
  - The Stability of a System

The Nature of a System

State

Input	Absence (Zero)	Presence
Absence (Zero)	No Response	Zero Input Response
Presence	Zero State Response	Complete Response

- System တစ်ခု၏ Input နှင့် State များပေါ်မူတည်ပြီး System တစ်ခု၏ Nature ကို သိရှိနိုင်ပါသည်။
- ရေတိုင်ကီအလွတ်တစ်လုံး (Zero State) ကိုရေဖြည့်မည် (Non Zero Input) ဆိုပါက ကျွန်တော်တို့ ရေပြည့်ဖို့ ကြာချိန်ကို သိနိုင်ပါသည်။ ရေတိုင်ကီကြီးလေ ပိုကြာလေ ဖြစ်မည်။
- ထို့အတူ ရေထည့်ထားသော ရေတိုင်ကီ (Non Zero State) မှ ရေထပ်မဖြည့်ဘဲ (Zero Input) ရေထုတ်မည်ဆိုပါက ကျွန်တော်တို့ ရေထွက်အားကို သိနိုင်မည်ဖြစ်သည်။ ရေတိုင်ကီကြီးလေ ရေထွက်အားကောင်းလေ ဖြစ်သည်။ တစ်နည်းအားဖြင့် Input သာမရှိရင် Output သည် System တစ်ခု၏ Nature ကို ဖော်ပြသော Natural Response တစ်ခု ဖြစ်သည်။
- Complete (Total) Response သည် Zero State Response နှင့် Zero Input Response တို့၏ Linear Combination ဖြစ်သည်။
- သို့သော် ကျွန်တော်တို့သည် Zero Input Response လို့ ခေါ်တဲ့ Natural Response ကို ပိုစိတ်ဝင်စားပါသည်။ ဘာကြောင့်လို့ ထင်ပါလဲ။

Homogenous Solutions (Zero Input Response)

$$(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n) y(t) = 0$$

- တကယ်လို့သာ Input သာ Zero ဆိုရင် Linear Differential Equation သည် Zero နှင့် တူသွားမှာပါ။ ဒါကို Linear Algebra မှာ Homogenous Equation လို့ခေါ်ပါသည်။
- ဒါဆိုရင် ကျွန်တော်တို့ သဘောပေါက်သွား လောက်ပါပြီ။ Natural Response သည် အမှန်တော့ Homogenous Solutions ဖြစ်ပြီးတော့ Forced Response သည် Particular Solutions ဖြစ်သည်။
- Linear Differential Equation ၏ Homogeneous Solution သည် ဘာဖြစ်မလဲ။
- ကျွန်တော်တို့ Linear Algebra မှာပြောခဲ့ပါသည်။ Derivative of Exponential သည် Eigen Equation တစ်ခု ဖြစ်သည်။ ဒါအပြင့် Linear လဲဖြစ်သည်။

$$\frac{d}{dx} e^x = e^x$$

- ဒါဆိုရင် $y(t) = ce^{\lambda t}$ လို့ယူဆကြည့်ရင် ဘယ်လို ဖြစ်မလဲ။

$$\begin{aligned} D^0 y &= ce^{\lambda t} \\ D^1 y &= c\lambda e^{\lambda t} \\ D^2 y &= c\lambda^2 e^{\lambda t} \\ &\dots \\ D^n y &= c\lambda^n e^{\lambda t} \end{aligned}$$

Characteristic Equations

$$c(a_0\lambda + a_1\lambda^1 + a_2\lambda^2 + a_3\lambda^3 + \dots + a_n\lambda^n)\, e^{\lambda t} = 0$$

- $y(t) = ce^{\lambda t}$ ကို ပြန်အစားထိုးကြည့်ရင် အပေါ် Equation ကို ရရှိမှာ ဖြစ်ပြီး $e^{\lambda t}$ ဖြင့် ပြန်စားလိုက်သည့် အခါ အောက် Equation ကို ရရှိမှာ ဖြစ်သည်။

- အောက် Polynomial Equation ကို Characteristic Equation ဟု ခေါ်သည်။ ထို့ Polynomial Equation ကိုရှင်းခြင်းဖြင့် Natural Response ကို ရရှိမှာ ဖြစ်ပါသည်။

$$(a_0\lambda + a_1\lambda^1 + a_2\lambda^2 + a_3\lambda^3 + \dots + a_n\lambda^n) = 0 \quad , \text{ since } e^{\lambda t} \neq 0$$

- Homogenous Solutions များကို ရလျှင် ကျွန်တော်တို့ Particular Solutions များရှာရမည် ဖြစ်သည်။

Particular Solutions

$$(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n) y(t) = f(t)$$

- Particular Solutions ရှာဖို့ အတွက် y(t) ကို Input နှင့် Form ခြင်းတူသည်ဟု ယူဆပြီး တွက်ပါသည်။

$$y(t) = cf(t)$$

- ပြီးလျှင် Linear Differential Equation မှာ အစားပြန်သွင်းပြီး Particular Solution များကို ရှာပါသည်။ အမှန်တော့ ဒါကို လက်တွေ့ တွက်ကြည့်ရင်တော့ တော်တော်လေး လက်ဝင်ပါသည်။ သို့သော်လည်း ဒီလိုတွက်တာထက် ပိုအဆင်ပြေသော နည်းလမ်းရှိပါသေးသည်။ ဒါကြောင့် ဒီနည်းလမ်းကို လက်တွေ့ သိပ်မသုံးပါ။

- နောက်ဆုံး Particular Solutions များကို ရရှိလာလျှင်တော့ Particular Solutions များနှင့် Homogenous Solutions များကို ပေါင်းခြင်းအားဖြင့် System တစ်ခုရဲ့ Complete Solution ကို ရရှိမှာ ဖြစ်ပါသည်။

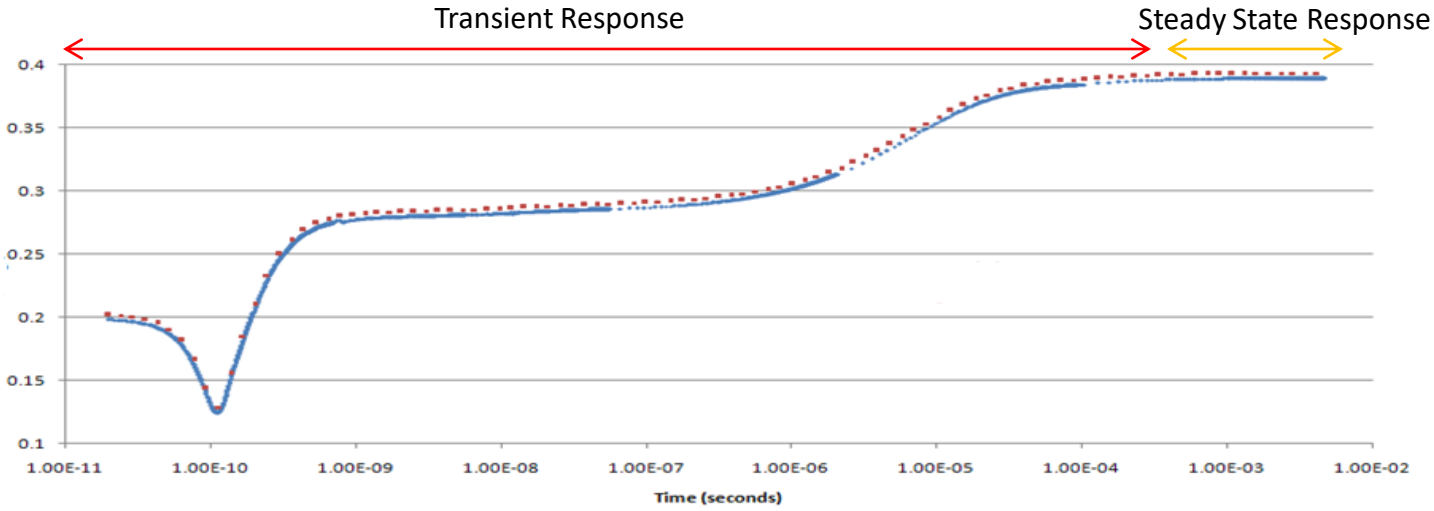
The Stability of a System

State

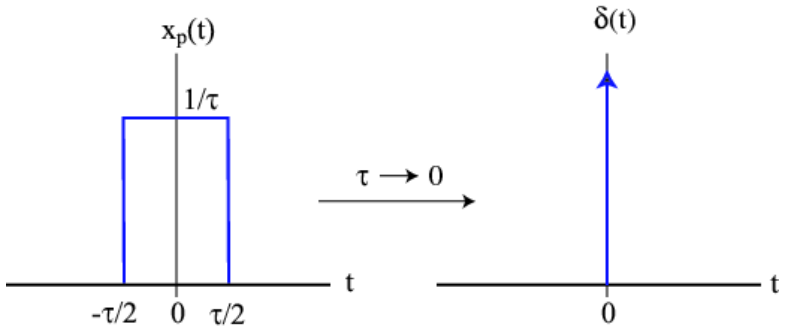
Input	Short Time	Long Time
Present	Transient Response	Steady State Response

- အမှန်တော့ System တစ်ခုရဲ့ Nature သာမက သူ့ရဲ့ Stability ကိုပါ ကျွန်တော်တို့ စိတ်ဝင်စားပါသည်။ Transient Response က Short Period Time အတွင်းမှာ System Response ဘယ်လို ရှိမလဲကို ပြပြီး Steady State Response က Long Period Time မှာ System Response ဘယ်လို ရှိမလဲကို ပြပါသည်။

- တကယ်လို့ System တစ်ခုက Steady State ကိုမရောက်လာရင် System က Stable မဖြစ်ပါ။ ဥပမာ၊ ဒေသတစ်ခု အတွင်း မွေဖွားနှုန်းနှင့် သေနှုန်းရှိသည်။ တကယ်လို့ ထိုဒေသ၏ လူဦးရေသည် Stable ဖြစ်ဖို့ ဆိုရင် မွေးနှုန်းနှင့် သေနှုန်းတူသွားရမည် ဖြစ်သည်။ သို့မဟုတ်ပါက လူဦးရေအလွန်တိုးလာခြင်း၊ အလွန်လျော့သွားခြင်းများ ဖြစ်ပါမည်။



Impulse Signal



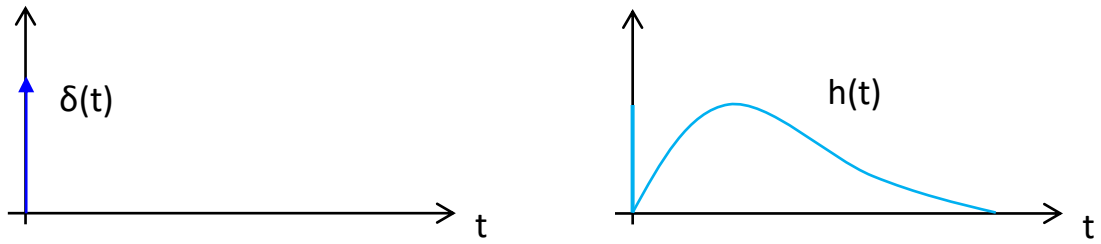
- Impulse တစ်ခုသည် အလွန်တိုသော အချိန်အတွင်း Energy အများကြီး ပေးလိုက်ခြင်း ဖြစ်သည်။ ဥပမာ၊ သစ်သားတုံးကို ခေါက်ခြင်း၊ ဗုံတီးခြင်း စသည်တို့ ဖြစ်သည်။

$$\int_{-\infty}^{\infty} \delta(t)dt = 1$$

- Unit Impulse တစ်ခု၏ Signal Energy (Strength) သည် အမြဲ 1 Unit ဖြစ်သည်။ ဒီအချက်သည် အင်မတန် အရေးကြီးပါသည်။

Impulse Response

$$(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n) y(t) = x(t) (b_0 D^0 + b_1 D^1 + b_2 D^2 + b_3 D^3 + \dots + b_m D^m)$$



- ဆိုပါတော့ t = 0 မှာ System တစ်ခုကို Impulse တစ်ခုပေးလိုက်မယ်ဆိုပါတော့။ t > 0 မှစပြီး Impulse မရှိတော့သည့် အတွက် ရလာသော Response သည် Natural Response နှင့် ဆင်တူမည် ဖြစ်သည်။

- ဒါဆိုရင် t = 0 မှာ ဖြစ်တဲ့ Zero State Response ကတော့ Impulse ၏ Form အတိုင်း ဖြစ်မည်။ ထို့ကြောင့် t=0, $h(t) = b_m \delta(t)$ ဖြစ်သည်။

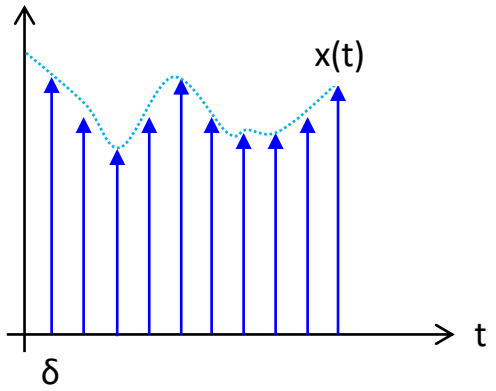
- တဖန် t = 0+ (Impulse ပျောက်သွားစ အချိန်လေးတွင်), System ၏ State သည် 1 ဖြစ်မည်၊ ဘာလို့လဲဆိုတော့ Unit Impulse ကရလာသော Energy သည် 1 Unit ဖြစ်၍ ဖြစ်သည်။ ထို့ကြောင့် System ၏ ပထမဆုံး နောက်နာ: (t = 0+)က State ကိုကိုယ်စားပြုသော $(D^{n-1})y(t)$ သည် 1 ဖြစ်၍ ကျန်တာများသည် Zero ဖြစ်သွားမည် ဖြစ်သည်။

- ထို့ကြောင့် t >= 0 မှာ, $y_h(t)$ = Homogenous Solutions with Initial Condition $(D^{n-1})y(t) = 1$ and the rest is zero ဖြစ်လျှင်

$$h(t) = b_m \delta(t) + [(b_0 D^0 + b_1 D^1 + b_2 D^2 + b_3 D^3 + \dots + b_m D^m) y_h(t)] u(t)$$

$b_m = 0$ if $m < n$, $u(t)$ = arbitrary

Signal In Terms of Impulses

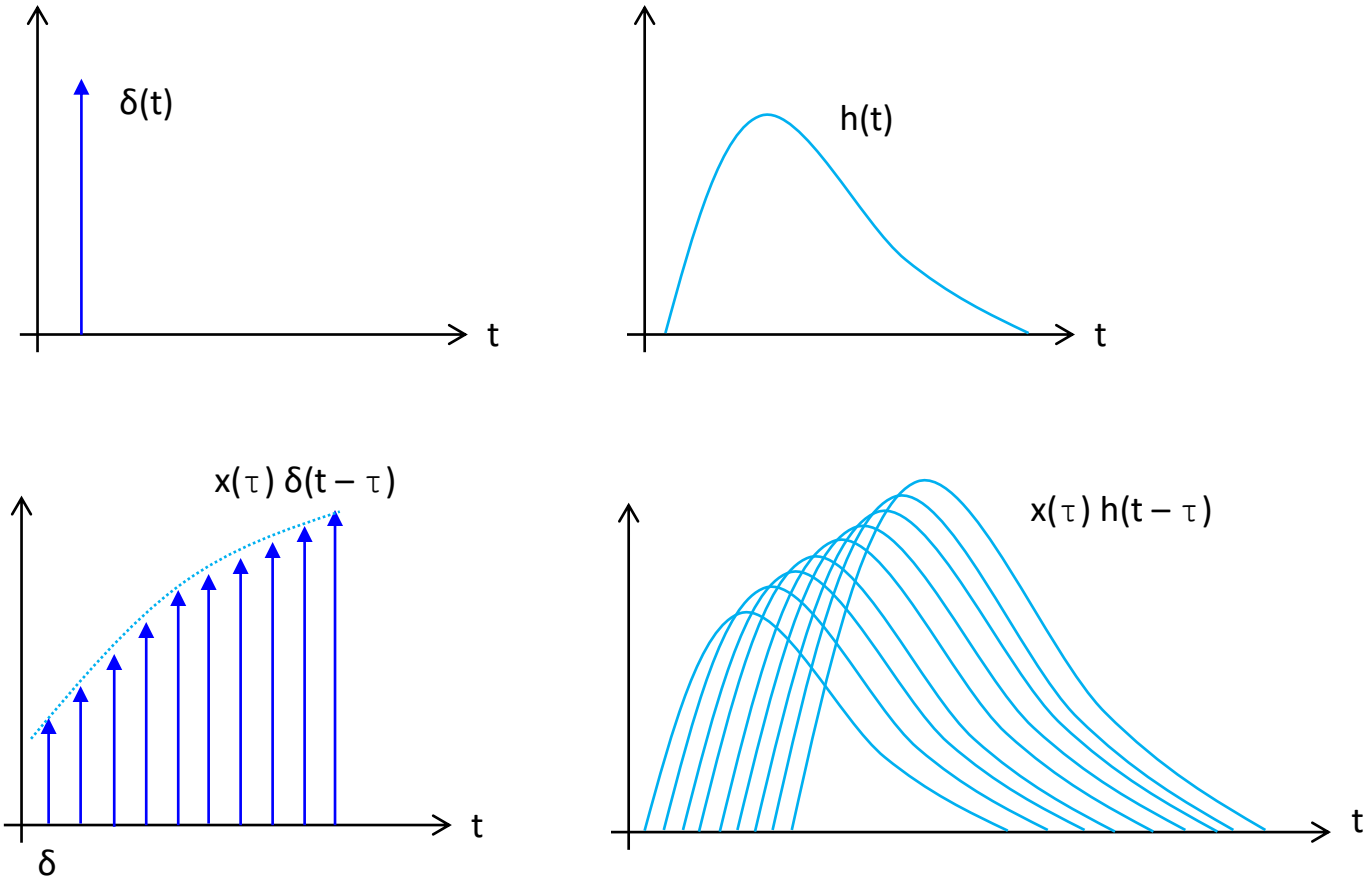


$$x(t) = \int_{-\infty}^{\infty} x(\tau)\delta(t-\tau)d\tau$$

- ကျွန်တော်တို့ Continuous Signal တစ်ခုကို Impulse များဖြင့် ဖော်ပြမည်ဆိုပါက အထက်ပါ Integral ကိုရရှိမည် ဖြစ်သည်။ Impulse များ Unit 1 ရှိသည့်အတွက် Signal တစ်ခုကို Unit Area 1 ရှိသော Rectangle များဖြင့် ဖွဲ့စည်းထားသည်ဟု ယူဆနိုင်ပါသည်။

- ဒါဆိုရင် ကျွန်တော့် Zero State Response များကို Input Signal In Terms of Impulse အနေဖြင့် စဉ်းစားကြည့်ရအောင်ပါ။

Zero State Response



- Impulse တစ်ခုရဲ့ Zero State Response သည် h(t) ဆိုပါက Signal In Terms of Impulse ရဲ့ Zero State Response သည် Response In Terms of Impulse Response h(t) ဖြစ်သည်။

Convolution

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

- Signal x(t) တစ်ခုအတွက် LTI System တစ်ခု၏ Zero State Response သည် Input Signal x(t) နှင့် LTI System ၏ Impulse Response တို့၏ Convolution ဖြစ်သည်။
- ဒါသည်ပင် အလွန်နာမည်ကြီးသော Convolution ၏ Definition ဖြစ်ပါသည်။ အမှန်တော့ Convolution ကို နေရာတကာနီးပါးမှာ သုံးပြီး Matrix Multiplication နီးပါးလောက် အသုံးဝင်သော Operation တစ်ခု ဖြစ်သည်။
- ထို့ကြောင့် ဒါကို သေသေချာချာ ဆွေးနွေးပါမည်။ ဥပမာ၊ Convolution ကို CNN (Convolutional Neural Network) များ၊ Image (Signal) Filter (Gaussian Blur, Beauty Filter)များတွင်လည်း သုံးသည်။
- ကျွန်တော်တို့ လက်တွေ့တွက်သည့်အခါ Discrete Convolution က ပိုအဆင်ပြေသည့်အတွက် Discrete Format ကိုပဲ စဉ်းစားပါမည်။

Discrete Signal

```
x[n] = [1,3,2,2,3,4,2,3,2,3, … ]
```

- အမှန်တော့ Single Discrete Signal များကို 1 Dimensional Array ဖြင့် ဖော်ပြလို့ ရပါသည်။ Multiple Discrete Signal (Image) များကို တော့ N Dimensional Array ဖြင့် ဖော်ပြကြပါသည်။

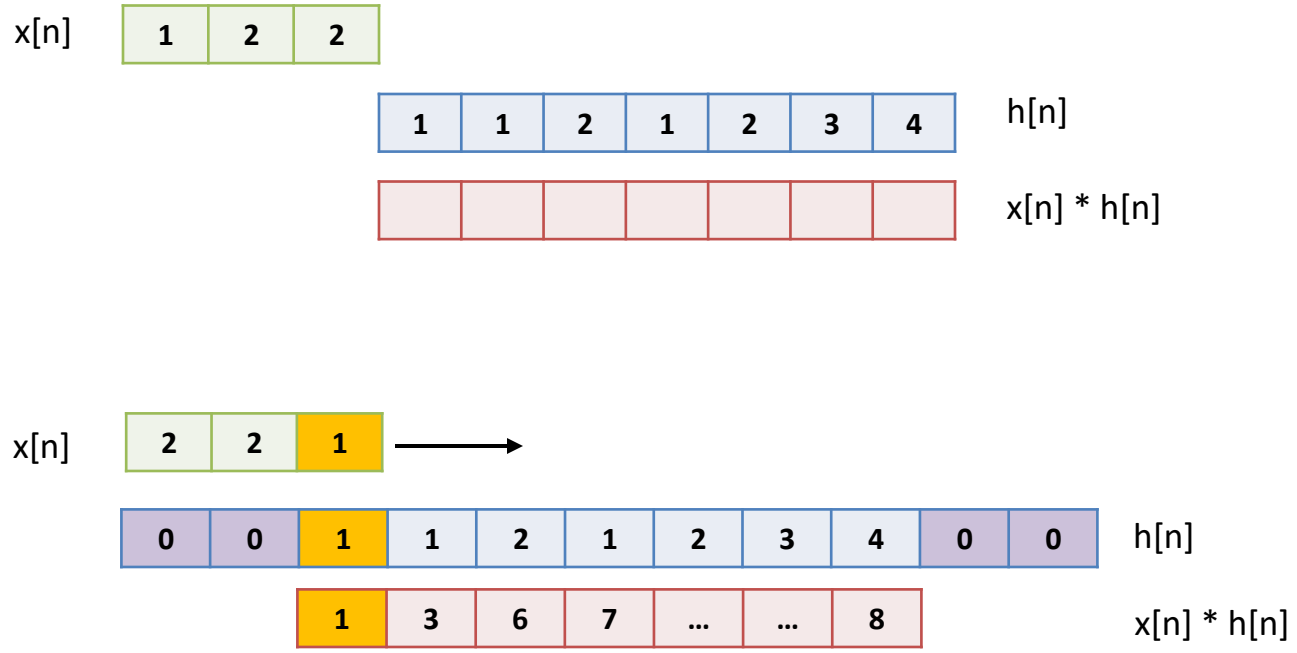
- အခုကျွန်တော်တို့ Discrete Convolution ကို စဉ်းစားကြည့်မှာ ဖြစ်ပါသည်။ အမှန်ကတော့ Convolution ကို Algorithm တစ်ခုအနေနဲ့ စဉ်းစားရင် မခက်ပါဘူး။

- သို့သော် Convolution ရဲ့ အဓိပ္ပါယ်က အင်မတန်လေးနက်ပါသည်။ ရှိသမျှ Linear System အားလုံးရဲ့ Zero State Response သည် Convolution ဖြစ်သည်။ ဒါကြောင့် သီအိုရီကို အရင်သွားတာ ဖြစ်ပါသည်။

- ဥပမာ၊ Linear Combination သည် အမှန်တော့ Sum of Products မျှသာ ဖြစ်သည်။ သို့သော် သီအိုရီအရ ဒါသည်ပင် Linear System အားလုံး၏ Linear Space ကိုယ်စားပြုသည်။ y = mx + b သည် အင်မတန် ရိုးရှင်းပါသည်။ သို့သော် အဓိပ္ပါယ် အင်မတန် နက်ရှိုင်းသည်။ ဘာလို့ဆိုတော့ Additivity နှင့် Scalability ကို ပြောလို့ ဖြစ်သည်။ အမှန်တော့ Linear ဖြစ်တဲ့ Entity များမှာသာ Scalability ရှိတာ ဖြစ်ပါသည်။ ထားပါတော့၊ ခင်ဗျား ပစ္စည်း 1 ခုရောင်းရင် 1 ကျပ်မြတ်တယ်၊ အခု 1 သန်း ရောင်း 1 သန်းမြတ်မလား။ မမြတ်ပါ။ Scalable မဖြစ်လို့ (တစ်နည်း Linear မဖြစ်လို့) ဖြစ်သည်။ ပစ္စည်းများလာလေလေ သူ့ရဲ့ ကုန်ကျစရိတ်လည်း တက်လာလေ ဖြစ်သည်။

- ကျွန်တော်တို့ အခု Convolution ကို Algorithm တစ်ခုအနေနဲ့ စဉ်းစားကြည့်မှာ ဖြစ်ပါသည်။

1 Dimensional Discrete Convolution



- ပထမဆုံး `x[n]` ကို `Flip (Reverse)` လုပ်ပါသည်။ ထို့နောက် `h[n]` ရဲ့ `Head` နဲ့ `Tail` ကို `Zero Padding` လုပ်ပါသည်။
- `x[n]` တဖြည်းဖြည်း `Slide` လုပ်လာပြီး `x[n]` နှင့် `h[n]`တို့ရဲ့ `Cell` နှစ်ခု တူတဲ့နေရာများရဲ့ `Linear Combination (Sum of Products)` ကိုရှာပြီး `Result` ရဲ့ `Cell` မှာထည့်ပါသည်။

2 Dimensional Discrete Convolution

x[n]

1	0	1
0	1	0
1	1	1

h[n]

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	1	2	1	2	0	0
0	0	1	2	3	1	1	0	0
0	0	2	2	1	2	2	0	0
0	0	1	2	1	2	1	0	0
0	0	2	1	1	2	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

x[n] * h[n]

1	2	…	…	2
1	4	…	…	…
…	…	…	…	…
…	…	…	…	…
2	…	…	…	1

Convolution Algorithm

```
function Convolute(kernel, array)

    var output = new Array[array.Rows, array.Columns];

    kernel = Flip(kernel);                      //Cross-Correlation does not need to flip.
    array = AddPadding(kernel, array);          //Need to add padding.

    for (x = 0; x < array.Rows - kernel.Rows; x++)
    {
            for  (y = 0; y < array.Columns - kernel.Columns; y++)
            {
                        var accumulator = 0;

                        for (i=0;  i < kernel.Rows; i++)
                        {
                                    for (j =0;  j < kernel.Columns; j++)
                                    {
                                                var row = x + i;
                                                var column  = y + j;

                                                accumulator += kernel[i, j] * array[row, column];
                                    }
                        }

                        output[x, y] = accumulator;
            }
    }
    return output;
}
```

Application of Convolutions

- Convolution သည် အမှန်တော့ LTI System အားလုံးရဲ့ Zero State Response များဖြစ်သည်။ ထို့ကြောင့် သူသည် နေရာတကာမှာပါပြီး Application များစွာ ရှိပါသည်။

- အထင်ရှားဆုံးကတော့ Digital Filter များဖြစ်သည်။ ကျွန်တော်တို့ရဲ့ Photoshop မှာသုံးတဲ့ Image Processing Operation အများစု သည် Convolution များဖြစ်ပါသည်။

- Image တစ်ခုကို Sobel Filter (G_x) နှင့် Convolution ရှာလိုက်ရင် ပုံရဲ့ Gradient in the Direction of X ကိုရမည် ဖြစ်သည်။ တစ်နည်းအား ပုံတစ်ခု၏ Edge များသည် Gradient အများဆုံးနေရာများ ဖြစ်သည်။ စဉ်းစားကြည့်ပါ မြေပုံမှာ ဆိုရင် ချောက်ကမ်းပါးများ ရှိသောနေရာများသည် Edge များဖြစ်သည်။

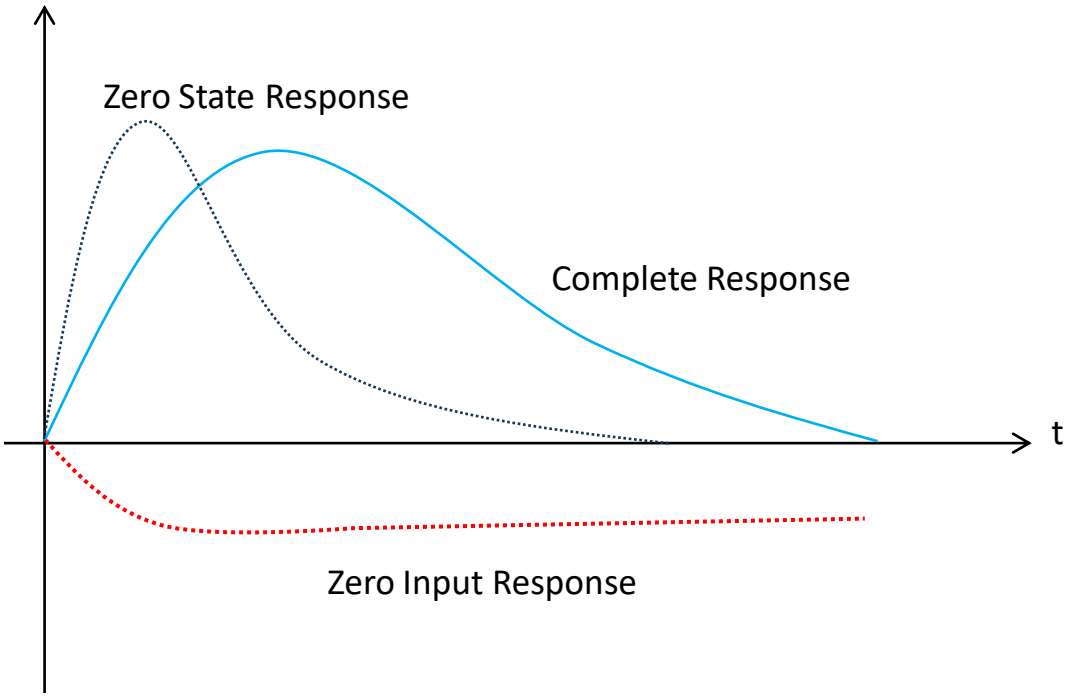
- ဒီအပိုင်းကို ကျွန်တော်တို့ Signal Processing Lab ကျမှ အသေးစိတ်ပြောပါမည်။ အခုတော့ Intro လောက်ပါ။ ကျွန်တော်တို့ အခုလောလောဆယ်တော့ သီအိုရီများကို အဓိက ထားရအောင်ပါ။

*

-1	0	1
-2	0	2
-1	0	1

=

Complete Response



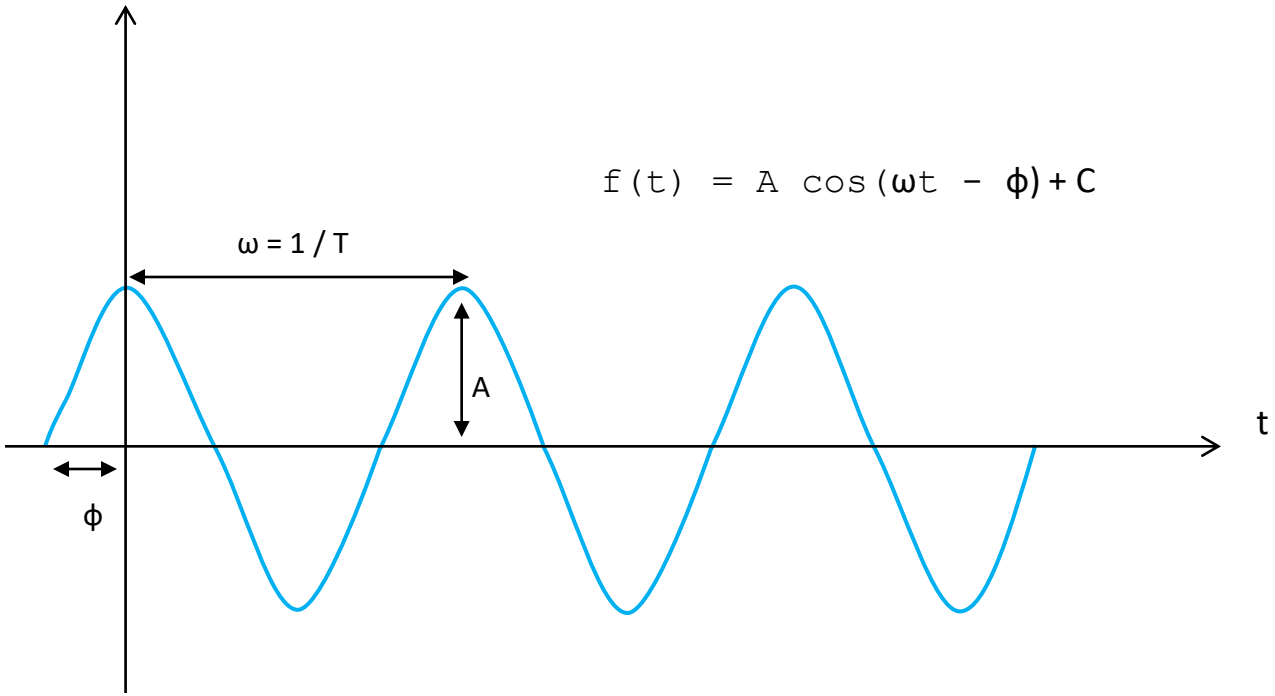
Zero Input Zero State

$$\text{Total (Complete) Response} = \sum_{i=1}^{n} c_i e^{\lambda_i t} + x(t) * h(t)$$

Time Domain Analysis

- အခု ကျွန်တော်တို့ Signal များကိုရော Signal များကို Process လုပ်တဲ့ Systems များကိုပါ နဲနဲ နားလည်သွားလောက်ပါပြီ။
- သို့သော် အခုထိ ကျွန်တော်တို့ ဆွေးနွေးတာသည် Time Domain အတွင်းမှ ဖြစ်သည်။
- Signal များကို Digital Signal, Analog Signal စသည်တို့ ခွဲခြားခြင်းသည် Time Domain အတွင်းမှ ဖြစ်သည်။ ထို့အတူ Systems များကိုလည်း System Response များကို လေ့လာခြင်းသည်လည်း Time Domain အတွင်းမှ ဖြစ်သည်။
- အခု ကျွန်တော်တို့ Frequency Domain ကို ဆက်လက် ဆွေးနွေးကြပါစို့။

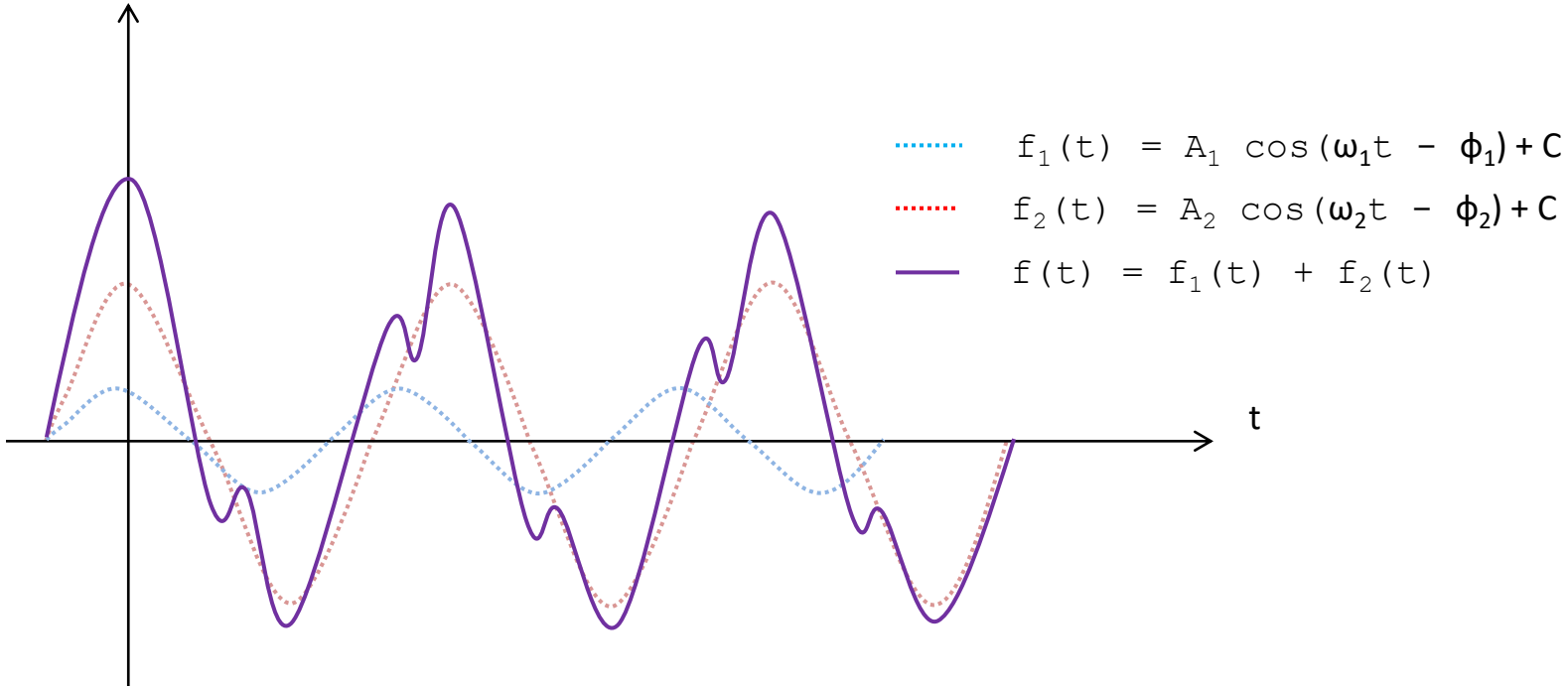
General Cosine Function



- ကျွန်တော်တို့ အခု Periodic Signal တွေအကြောင်း ပြောပါမည်။ အဲဒါကို မပြောခင် General Cosine Function တစ်ခုကို ဒီလို ဖော်ပြနိုင်ပါသည်။

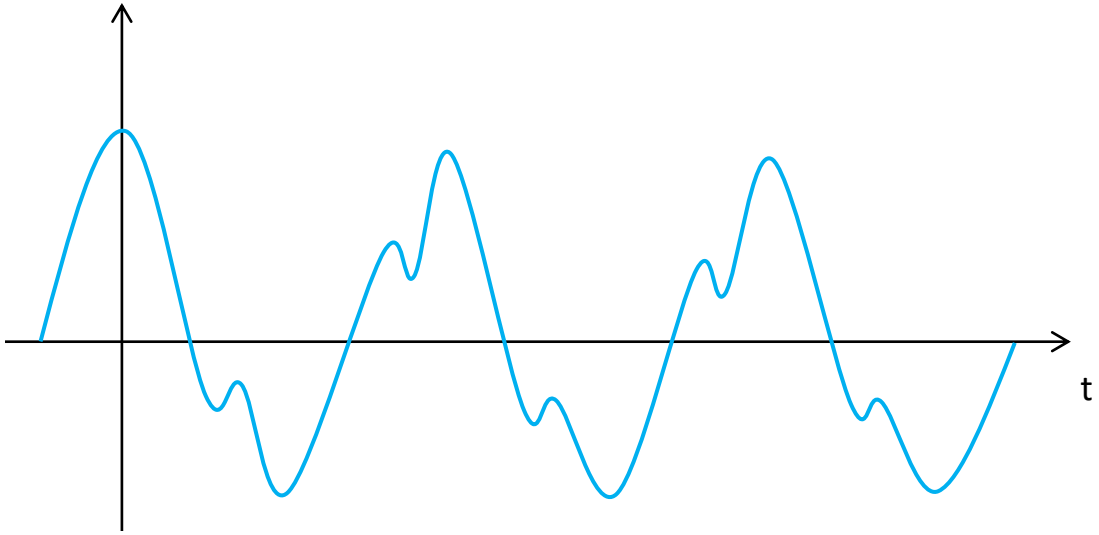
- General Cosine Function သည် Periodic Signal ထဲမှာ အရိုးရှင်းဆုံးသော Signal တစ်ခု ဖြစ်ပါသည်။

Periodic Signal



- ကျွန်တော်တို့ ပုံစံမတူတဲ့ Cosine Function နှစ်ခုကို ပေါင်းခြင်းအားဖြင့် ထူးခြားတဲ့ Periodic Signal တစ်ခုကို ရရှိပါသည်။
- ဒါဆိုရင် ကျွန်တော်တို့သည် ကြိုက်တဲ့ Periodic Signal တစ်ခုကိုရော Cosine Function များပေါင်းခြင်းအားဖြင့် ရရှိနိုင်မလား။
- ဒီမေးခွန်းရဲ့ အဖြေကို Joseph Fourier ဆိုတဲ့ ပညာရှင်က ရှာဖွေတွေ့ရှိခဲ့ပါသည်။

Fourier Series



$$x(t) = C + \sum_{k=1}^{\infty} A_k \cos\left(k\omega_0 t - \phi_k\right)$$

- Fourier Series အရ အချက် 2 ချက်သာကိုက်ညီရင် ကျွန်တော်တို့သည် ကြိုက်တဲ့ Periodic Signal တစ်ခုကို Sine Function များပေါင်းခြင်းအားဖြင့် ရရှိနိုင်ပါသည်။

- Periodic Signal တစ်ခုမှာ Primary Cosine Function ရဲ့ Fundamental Frequency (ω_0) တစ်ခု ရှိရမည် ဖြစ်ပြီး အခြား ထပ်ပေါင်းမည့် Cosine Function များရဲ့ Frequency များသည် Fundamental Frequency ၏ Integer (Harmonic) Multiple ($k\,\omega_0$) ဖြစ်ရမည် ဖြစ်သည်။

- တစ်နည်းအားဖြင့် Periodic Signal တစ်ခုသည် Linear Combination of Harmonic Cosine Function များဖြစ်ပြီး ပထမဆုံး Cosine Function ၏ Frequency သည် Fundamental Frequency (ω_0) ဖြစ်သည်။

Fourier Series

$$x(t) = C + \sum_{k=1}^{\infty} A_k \cos(k\omega_0 t - \phi_k)$$

Since $\cos(a - b) = \cos(a)\cos(b) + \sin(a)\sin(b)$,

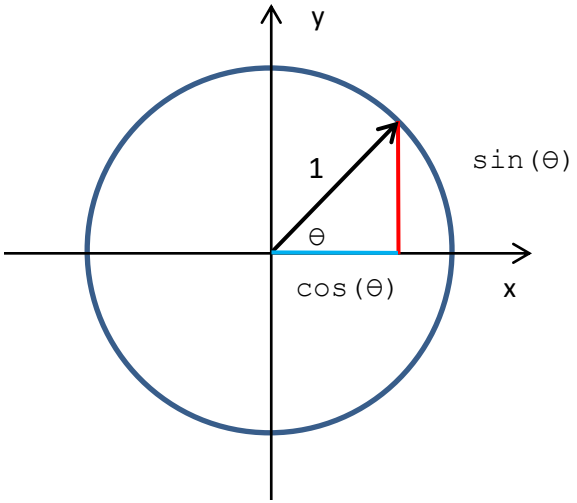
$$x(t) = C + \sum_{k=1}^{\infty} a_k \cos(k\omega_0 t) + b_k j\sin(k\omega_0 t)$$

$$a_k = A_k\cos(\phi_k) \ , \ b_k = A_k\sin(\phi_k)$$

- ဒါကို **Fourier Series** ရဲ့ ပိုပြီး အရေးကြီးတဲ့ **Form** ဖြစ်ပါတယ်။ ကျွန်တော်တို့ ϕ_k ကို ဖယ်လိုက်ပြီး **Cosine** နှင့် **Sine Function** များဖြင့် ဖော်ပြပါသည်။
- ဒီ **Form** က ဘာလို့ ပိုအရေးကြီးလဲ ဆိုတာကို ဆက်ပြီး ဆွေးနွေးပါမည်။

Orthogonal Signal Space

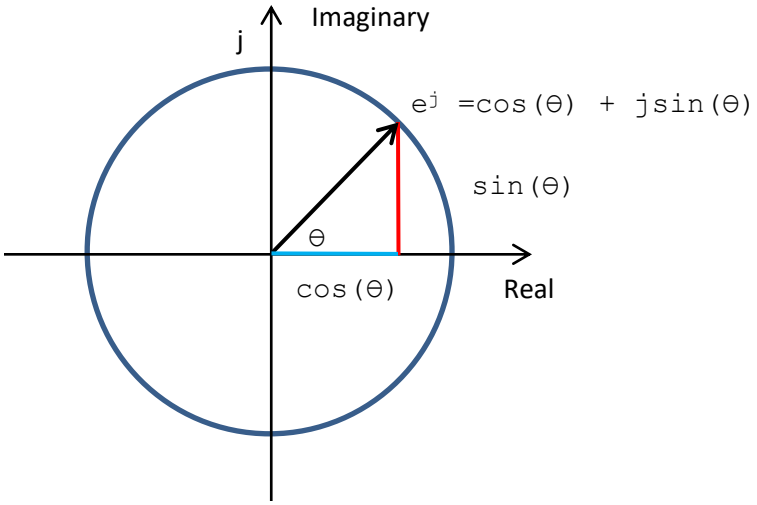
- Linear Algebra မှာ Orthogonal Basis များ အကြောင်းပြောခဲ့ပါသည်။ Orthogonal Basis များသည် တစ်ခုကို တစ်ခုကို Orthogonal (ထောင့်မှန်ကျပြီး) Orthogonal Basis များ၏ Linear Span သည် Vector Space တစ်ခုကို ကိုယ်စားပြုပါသည်။ အမှန်တော့ $\sin(\Theta)$ နှင့် $\cos(\Theta)$ တို့သည် Orthogonal ဖြစ်ပါသည်။



- ဒါဆိုရင် Orthogonal ဖြစ်သော Sine နှင့် Cosine တို့သည် Orthogonal Basis များဖြစ်ပြီး Sine နှင့် Cosine တို့၏ Linear Span သည် Vector Space တစ်ခု ဖြစ်သည်။

- ထို့ကြောင့် Fourier Series ဖြင့် ဖော်ပြနိုင်သော Periodic Signal အားလုံးသည် Sine နှင့် Cosine တို့၏ Vector Space အတွင်းရှိ Vector များသာ ဖြစ်သည်။

Euler's Formula



$$e^{j\theta} = \cos(\theta) + j\sin(\theta) \qquad j = \sqrt{-1}$$

- Euler's Formula သည် Exponential Function, Trigonometric Function နှင့် Complex Plane (Number) များကို လှပစွာ ဆက်သွယ်ပေးပါသည်။
- ဒီထက်ပို အရေးကြီးသည် Linear System များ၏ Homogenous Solutions များသည် Exponential Function များဖြစ်ပြီး Periodic Signal များကိုလည်း Cosine နှင့် Sine Function များဖြင့် ဖော်ပြလို့ ရသည်။ ဒီဟာက တော်တော်လေးကို စိတ်ဝင်စားဖို့ ကောင်းပါသည်။

Important Identities

$$\frac{d}{dx}e^x = e^x$$

$$\frac{d}{dx^2}\sin(x) = -\sin(x)$$

$$\frac{d}{dx^2}\cos(x) = -\cos(x)$$

$$\cos(\Theta) = \frac{e^{j\theta} + e^{-j\theta}}{2}$$

$$\sin(\Theta) = \frac{e^{j\theta} - e^{-j\theta}}{2}$$

General Fourier Series

$$x(t) = C + \sum_{k=-\infty}^{\infty} a_k \cos(k\omega_0 t) + b_k j\sin(k\omega_0 t) = C + \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0}$$

- General Fourier Series ကို ကျွန်တော်တို့ Euler's Formula ဖြင့် ဖော်ပြနိုင်ပြီး Orthogonal Signal Space သည် Complex Orthogonal Signal Space ဖြစ်သွားမည် ဖြစ်သည်။ Complex Number နှင့် Plane များကိုတော့ ကျွန်တော်တို့ မဆွေးနွေးတော့ပါဘူး။
- Complex Plane တွင် Conjugate Pair များရှိသည် ဖြစ်၍ $-\infty$ မှ ∞ အထိရှိမည် ဖြစ်သည်။
- ကျွန်တော်တို့ နောက်ပိုင်းမှာ General Fourier Series ကိုသာ ဖော်ပြတော့မှာ ဖြစ်ပါသည်။
- ဒီနေရာမှာ ကျွန်တော်တို့ သတိထားရမည်က Fourier Series သည် Infinite Series တစ်ခု ဖြစ်သည်။ သို့ဆိုလျှင် အချို့သော Periodic Signal များ၊ ဥပမာ Periodic Digital Signal များကို ဖော်ပြရာတွင် Infinite Sinusoidal (Sine and Cosine) Component များလိုအပ်မည် ဖြစ်သည်။

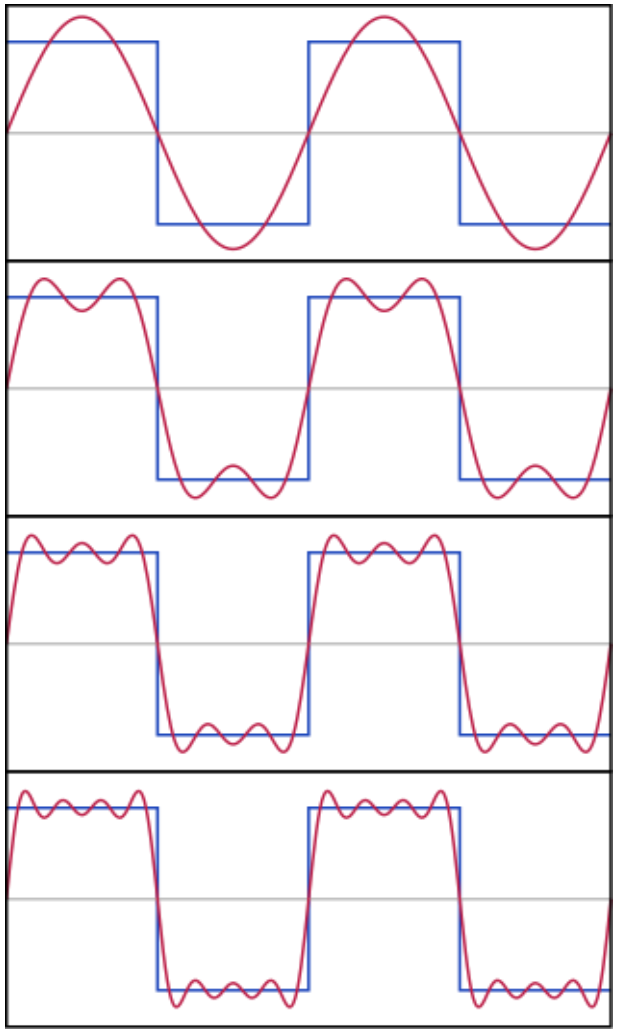
Periodic Digital Signal

$$x(t) = \frac{\pi}{4} \sum_{k=1,3,\ldots}^{\infty} \frac{1}{k} \sin(k\omega_0 t)$$

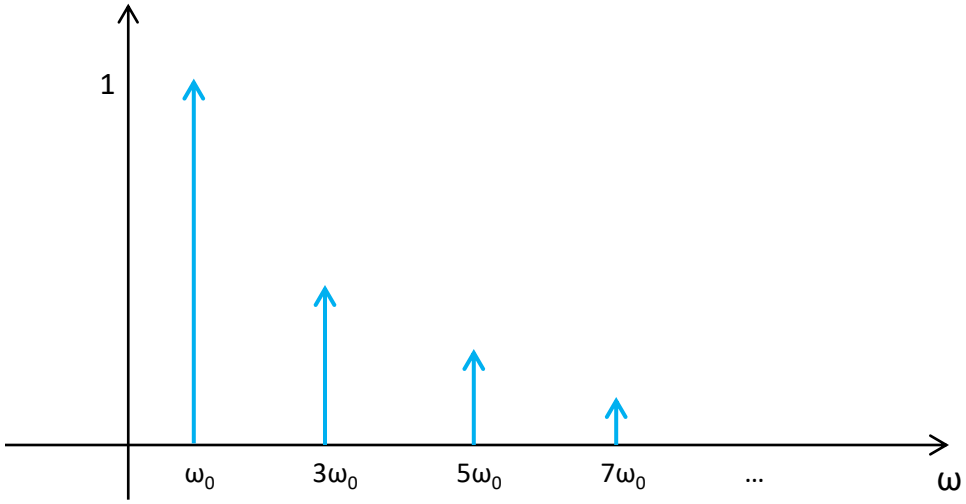
- Periodic Digital (Square) Signal တစ်ခုကို Fourier Series ဖြင့် ဖော်ပြမည် ဆိုပါက ကျွန်တော်တို့ Infinite Sinusoidal (Sine and Cosine) Component များလိုအပ်ပါသည်။
- တစ်နည်းအားဖြင့် Periodic Digital Signal တစ်ခုသည် Infinite Number of Sinusoidal Component များပါဝင်သည်ဟု ဆိုလိုပါသည်။
- ဒါဆိုရင် Periodic Digital (Square) Signal တစ်ခုမှာ Infinite Energy ရှိသည်ဟု ဆိုလိုရာရောက်နေသည်။
- ဒါကြောင့် ဒါကို သေသေချာချာ စဉ်းစားဖို့ လိုအပ်လာပါသည်။ ဒီတစ်ခါ ကျွန်တော်တို့ Fourier Series ၏ Coefficient များကို စဉ်းစားကြည့်ပါမည်။
- Periodic Digital (Square) Signal တစ်ခု၏ Coefficient သည် 1/k ဖြစ်သည်။ သို့ဖြစ်၍

$$\lim_{k \to \infty} \frac{1}{k} = 0$$

- ထို့ကြောင့် Periodic Digital Signal တစ်ခုသည် Infinite Number of Sinusoidal Component များပါဝင်သော်လည်း တဖြည်းဖြည်း Zero ဖြစ်သွားမည်။



Fourier Spectrum



$$x(t) = \frac{\pi}{4} \sum_{k=1,3,\ldots}^{\infty} \frac{1}{k} \sin(k\omega_0 t)$$

- အမှန်တော့ အရှင်းဆုံးပြောရရင် Fourier Spectrum သည် Fourier Series မှာပါဝင်သော Harmonic Frequency ၏ Fourier Coefficient များကို ဖော်ပြခြင်း ဖြစ်သည်။

- သို့သော် Fourier Series တွင် ပါဝင်သော Signal များသည် Sine နှင့် Cosine ဆိုပြီး Component 2 ခုပါဝင်သော Vector များဖြစ်ကြသည်။ ထို့ကြောင့် Vector၏ Magnitude နှင့် Direction ကို ဖော်ပြရမည်။ Magnitude ကို ဖော်ပြသော Fourier Spectrum ကို Amplitude Spectrum ဟုခေါ်ပြီး Direction ကို ဖော်ပြသော Fourier Spectrum ကို Phase Spectrum ဟုခေါ်သည်။

- Periodic Signal များ၏ Fourier (Frequency) Spectrum များသည် Discrete ဖြစ်ကြသည်။

Fourier Synthesis

- Fourier Synthesis သည် Wave Form Generation (Signal Construction) များနှင့် ဆိုင်ပါသည်။
- လိုအပ်သော Periodic Signal များကို Fourier Series ကို သုံးပြီး Generate လုပ်ခြင်းဖြစ်သည်။
- တစ်နည်းအားဖြင့် လိုအပ်သော Periodic Signal များက Vector Signal Space ၏ Linear Span များ အဖြစ်ဖော်ပြခြင်းဖြစ်သည်။

$$x(t) = C + \sum_{k=-\infty}^{\infty} a_k \cos(k\omega_0 t) + b_k j\sin(k\omega_0 t) = C + \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}$$

Fourier Analysis

- Fourier Analysis ကတော့ Periodic Signal များကို ဘယ်လို Sinusoidal Component များဖြင့် ဖွဲ့စည်းထားသည်ကို လေ့လာခြင်း ဖြစ်သည်။
- အများအားဖြင့် Fourier Coefficient များကို ရှာခြင်း၊ Fourier Spectrum များကို လေ့လာခြင်းတို့ကို ပြုလုပ်ကြသည်။

$$c_k = \frac{1}{T_0}\int_{o}^{T_0} x(t)e^{-jk\omega_0 t}dt \qquad , T_0 = \text{Period and } c_k = |c_k|e^{j\Theta}$$

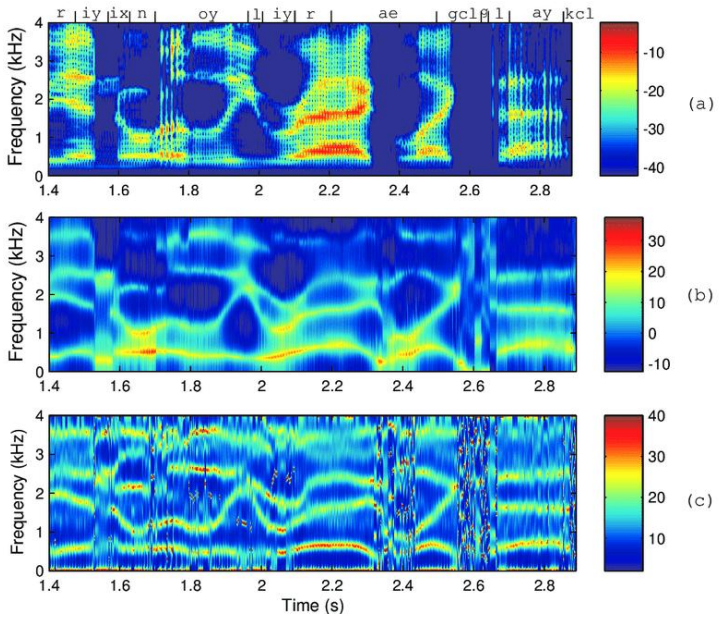
- အမှန်တော့ Signal များကို Frequency Component များနှင့် Coefficient များ အနေဖြင့် လေ့လာခြင်းသည်ပင် Frequency Domain Analysis ဖြစ်သည်။
- ထို့ကြောင့် Signal နှင့် System များကို Time Domain အနေဖြင့်ရော၊ Frequency Domain အနေဖြင့်ပါ လေ့လာနိုင်ပါသည်။
- သို့သော် အများအားဖြင့် Frequency Domain ကိုပိုပြီး ဦးစားပေးကြပါသည်။

Frequency Domain Analysis

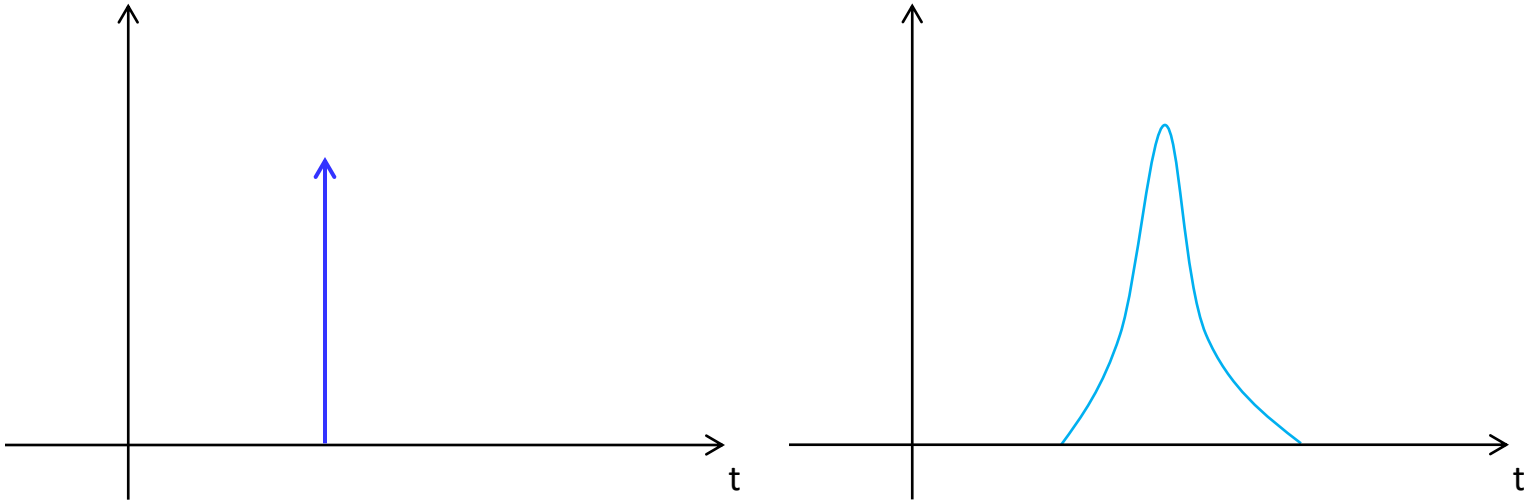
- ကျွန်တော်တို့ စကားပြောတဲ့ အခါ၊ သီချင်းနားထောင်တဲ့ အခါ အသံများ ကြားရပါသည်။ ဆူညံသံနှင့် ဂီတသံ ဘာကွာပါလဲ။ ဆူညံသံ သည် Random Signal များဖြစ်ပြီး ဂီတသံသည် Harmonic Signal များဖြစ်ပါသည်။ စကားပြောတဲ့ အခါမှာလည်း Vowel လို့ခေါ်တဲ့ သရသံကို ဘာသာစကားတိုင်း အခြေခံကြရပါသည်။ Vowel များသည် စကားသံများ၏ Fundamental Frequency များဖြစ်၍ ဖြစ်သည်။ ထို့ကြောင့် သီချင်းနားထောင်ခြင်း၊ စကားနားလည်ခြင်းသည် အမှန်တော့ Frequency Component များကို Process လုပ်ခြင်းတာ ဖြစ်ပါသည်။

- ထို့အတူ ကျွန်တော်တို့ မြင်သည့်အခါ အဝါ၊ အပြာ၊ ခရမ်း၊ မဲနယ်စသည်ဖြင့် မြင်ကြပါသည်။ အမှန်တော့ ထိုရောင်စဉ်များသည် Light Spectrum (Frequency) များသာ ဖြစ်ပါသည်။ ကျွန်တော်တို့၏ နား နည်းတူ မျက်စိသည်လည်း အမှန်တော့ Frequency Component များကို Process လုပ်ခြင်းတာ ဖြစ်ပါသည်။

- ဒါဆိုရင် Frequency များသည် ကျွန်တော်တို့ ထင်တာထက် လောကကြီးကို ပိုပြီး လွှမ်းမိုးနေပါသည်။ ထို့ကြောင့် ကျွန်တော်တို့ Signal Processing မှစပြီး AI, Machine Learning စသည်တို့ အထိ Frequency Analysis လုပ်ဖို့သည် ပိုပြီး အရေးကြီးလာပါသည်။

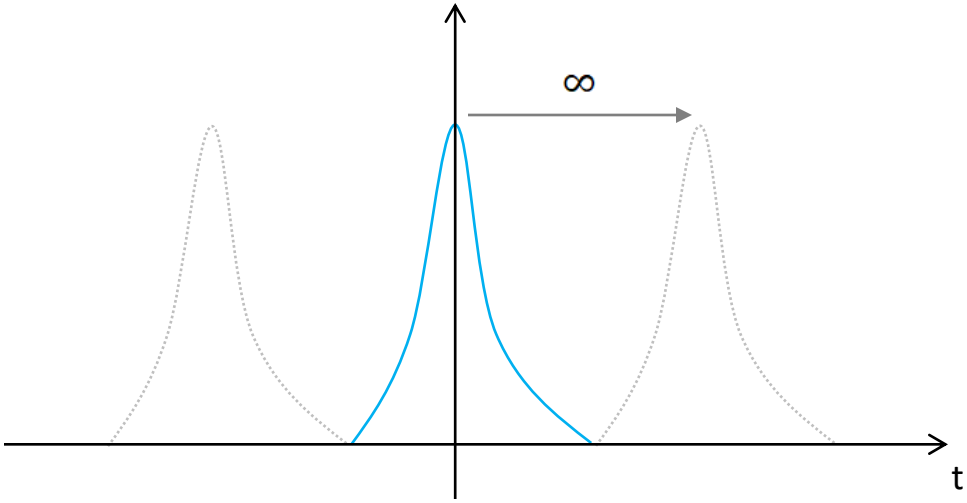


Non Periodic Signal



- Signal များအားလုံးသည် Periodic Signal များမဟုတ်ကြပါ။ ဥပမာ၊ Impulse Signal သည် Periodic Signal မဟုတ်ပါ။
- Non Periodic Signal များကို ရော Fourier Series ဖြင့် ဖော်ပြလို့ ရမလား။
- Non Periodic Signal များကတော့ Fourier Series ဖြင့် ဖော်ပြလို့ မရပါ။ သို့သော် Fourier Transform ဖြင့် ဖော်ပြလို့ ရပါသည်။
- အမှန်တော့ Fourier Transform သည် Periodic Signal များကို ရော၊ Non Periodic Signal များကိုပါ ဖော်ပြနိုင်ပါသည်။
- Fourier Transform မဆွေးနွေးခင် ကျွန်တော်တို့ Fourier Integral ကို ကြည့်ကြပါစို့။

Fourier Integral



- Periodic Signal များသည် Period တစ်ခုပြီးလျှင် Repeat ပြန်ဖြစ်လာသော Signal များဖြစ်သည်။

- တကယ်လို့သော Period တစ်ခုသည် Infinite ဖြစ်သွားမည်ဆိုပါစို့။ ထိုအခါ Periodic Signal တစ်ခုသည် Infinitely ကြာမှသာ Repeat ပြန်ဖြစ်မည် ဖြစ်သည်။ Infinitely ကြာမှ Repeat ဖြစ်မည်ဆိုလျှင် ဘယ်တော့မှ Repeat ပြန်မဖြစ်ဟု ဆိုလိုရာရောက်သည်။

$$\lim_{T_0 \to \infty} f\, T_0(t) = f(t)$$

$$f(t) = \sum_{k=-\infty}^{\infty} c_k\, e^{jk\omega_0 t}$$

- သို့သော် ကံကောင်းသည် Fourier Series သည် Infinite Series ဖြစ်၍ Signal တစ်ခု၏ Period တစ်ခုသည် Infinite ဖြစ်လဲ ပြဿနာ မရှိတော့ပါ။ အမှန်တော့ Infinite Period ပဲရှိရှိ၊ Finite Period ပဲရှိရှိ၊ Signal တစ်ခု၏ Frequency Component များကိုသာ ကျွန်တော်တို့က စိတ်ဝင်စားတာ ဖြစ်ပါသည်။

Fourier Transform

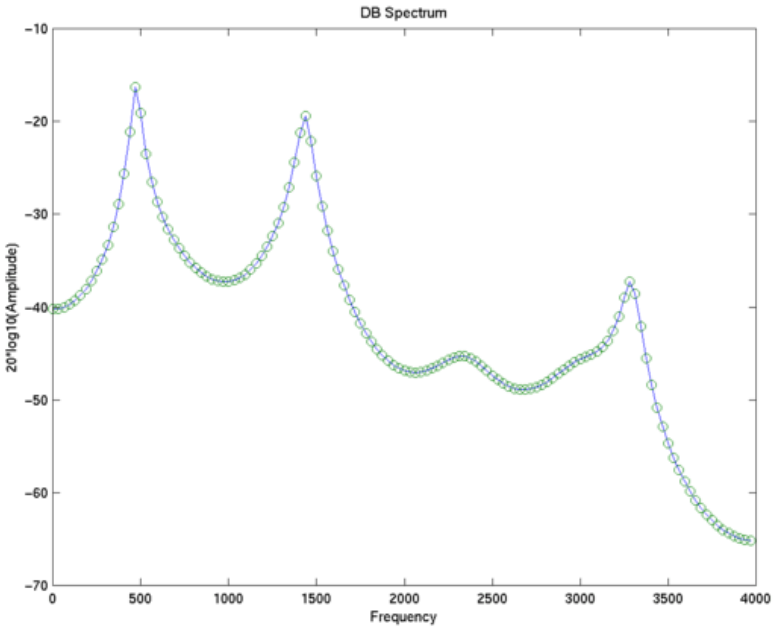
$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)\, e^{j\omega t} d\omega$$

- Fourier Transform သည် Signal (Infinite Period ပဲရှိရှိ၊ Finite Period ပဲရှိရှိ) တစ်ခုမှာ ပါဝင်သော Frequency Component များကို ဖော်ပြသော Continuous Function တစ်ခု ဖြစ်သည်။
- Fourier Transform သည် Continuous Function of Frequency ဖြစ်ပြီး $-\infty$ မှ ∞ အထိ Frequency Component အားလုံးပါဝင်ပါသည်။
- ဤသို့ဖြင့် Fourier Transform သည် Signal အားလုံးကို Time Domain မှ Frequency Domain သို့ပြောင်းပေးပါသည်။
- Periodic Signal များ၏ Frequency Spectrum သည် Discrete ဖြစ်ပြီး Non Periodic Signal များ၏ Frequency Spectrum သည် Continuous ဖြစ်သည်။
- Signal တစ်ခုတွင် Infinite Frequency Component များပါဝင်နိုင်သော်လည်း Infinite Energy မရှိပါသည်။ အချို့ Frequency Component များသည် တဖြည်းဖြည်းနှင့် Zero နားသို့ ချဉ်းကပ်သွားမည် ဖြစ်သည်။

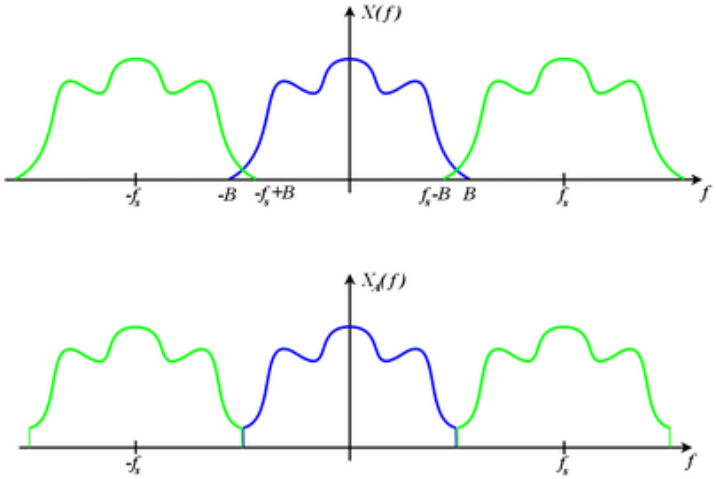
Bandwidth

- Bandwidth ဆိုတဲ့ စကားကို လူတော်တော်များများ ရင်းနှီးပြီးသား ဖြစ်မှာပါ။ Internet Broadband တို့၊ Wideband တို့ပေါ့။
- အမှန်တော့ Bandwidth ဆိုတာသည် Frequency Spectrum မှ လာတာ ဖြစ်ပါသည်။ Signal တစ်ခုမှာ Frequency Component များ Infinitely ပါနိုင်ပါသည်။ သို့သော်လည်း ဘယ် Signal မှ Infinite Energy မရှိပါ။
- ထို့ကြောင့် ဘယ် Frequency Component များမှာ Energy Level ဘယ်လောက်ရှိသည်ကို ဖော်ပြကြပါသည်။ ဒါသည် Amplitude Frequency Spectrum ဖြစ်ပါသည်။
- သို့သော် Energy Level များကို ဖော်ပြသည့်အခါ Logarithmic Scale ($20\log_{10}$) ကို သုံးပါသည်။ ဘာလို့လဲ ဆိုတော့ ကျွန်တော်တို့ နားသည် Logarithmic Scale ကိုသာ Sensitive ဖြစ်သည်။ ဥပမာ၊ အသံ 1 ဆ ပိုကျယ်သွားခြင်းသည် အမှန်တော့ 10 ဆ ပိုကျယ်သွားခြင်း ဖြစ်သည်။ အပြင်မှာ 10 ဆ ပိုကျယ်သွားသည်က နားက 1 ဆ ပိုကျယ်လို့သာ ထင်ရသည်။ ထို Logarithmic Scale ကို Decibel (dB) ဖြင့် ဖော်ပြကြပါသည်။
- Signal တစ်ခု၏ Bandwidth ဆိုတာသည် Signal တစ်ခု၏ Energy Level အမြင့်ဆုံး Frequency Component နှင့် Energy Level အနိမ့်ဆုံး Frequency Component တို့၏ ကွာခြားချက် ဖြစ်သည်။ ဥပမာ၊ ကျွန်တော်တို့ နားသည် 20Hz မှ 20 KHz အထိ Frequency Component များကို Sensitive ဖြစ်ပါသည်။ ဒါသည် နား၏ Bandwidth ဖြစ်သည်။
- ထို့ကြောင့် Bandwidth ကြီးလေလေ Frequency Component များလေလေ ဖြစ်သည်။

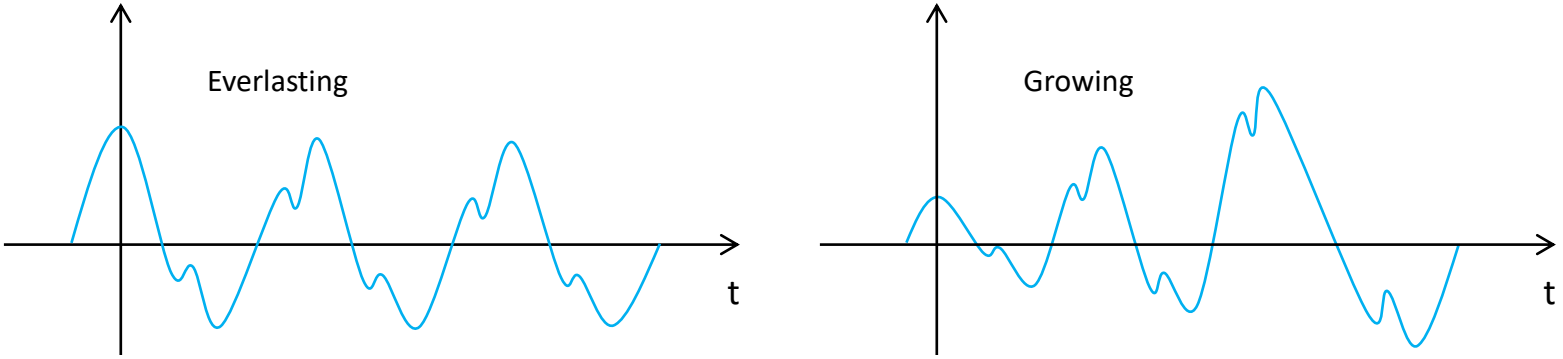


Nyquist Bandwidth

- အစပိုင်းမှာ ကျွန်တော်တို့ Continuous Signal များကို Discrete Signal များ အဖြစ်ပြောင်းရင် Nyquist-Shannon Sampling Theorem သုံးရကြောင်း ပြောခဲ့ပါသည်။
- Nyquist-Shannon Sampling Theorem အရ Sampling Rate သည် Fundamental Frequency ၏ 2 ဆ ဖြစ်ရမည်လို့ ပြောခဲ့ပါသည်။
- အခုကျွန်တော်တို့ ဒါကို ပိုပြီး သေသေချာချာ ဆွေးနွေးပါမည်။
- Signal တစ်ခု၏ Effective Bandwidth တစ်ခုသည် (B) ဖြစ်မည် ဆိုပါက Signal တစ်ခုကို Sample လုပ်ဖို့ (2B) Bandwidth လိုပါသည်။
- Effective Bandwidth ဆိုသည်က Signal တစ်ခု၏ Signal Energy အများစုရှိသော Bandwidth ဖြစ်သည်။
- ဥပမာ၊ လူ၏နားသည် 20Hz မှ 20 KHz အထိ Frequency Component များကို Sensitive ဖြစ်ပါသည်။ ထို့ကြောင့် နား၏ Effective Bandwidth = 20KHz – 20Hz = 19,980 Hz ဖြစ်သည်။
- ဒါကြောင့် ကျွန်တော်တို့က Audio များကို Sample လုပ်မည်ဆိုလျှင် 39,960 Hz လိုမည် ဖြစ်သည်။ ထို့ကြောင့် Audio File များစုကို 40 KHz နားမှာ ထားတာ ဖြစ်သည်။ ဥပမာ၊ wav file များကို 44,100 Hz with 16 bits per sample ဖြင့် Sample လုပ်တာ ဖြစ်သည်။



Laplace Transform



- ကျွန်တော်တို့ အခုအထိ Signal များကို ပြောခဲ့တာသည် Everlasting (Steady State) Signal ဟုခေါ်သော Signal (Periodic or Non Periodic) များဖြစ်သည်။ ဒါကြောင့် Fourier Integral သည် $-\infty$ မှ ∞ အထိ ဖြစ်သည်။

- တကယ်လို့ Signal တစ်ခုသည် Growing သို့မဟုတ် Decaying (Transient) ဖြစ်မည်ဆိုပါက Fourier Transform ဖြင့် မလုံလောက်တော့ပါ။ ထို့ကြောင့် ပိုပြီး General ဖြစ်သော Transform တစ်ခုကို သုံးကြပါသည်။ ထို Transform ကို Laplace Transform ဟုခေါ်ပါသည်။

- တစ်နည်းအားဖြင့် Laplace Transform သည် Fourier Transform ၏ Extension တစ်ခု ဖြစ်ပါသည်။

$$\boldsymbol{L}(s) = F(s) = \int_0^{\infty} f(t)e^{-st}dt \qquad \text{where } s = \sigma + j\omega$$

Zero State Frequency Response

$$F(s) = \int_0^{\infty} f(t)e^{-st}dt \qquad \text{where } s = \sigma + j\omega$$

- အခုကျွန်တော်တို့ System များ၏ Frequency Response များကို ဆွေးနွေးကြည့်ပါမည်။ Time Response များနည်းတူ System ၏ Frequency Response များတွင်လည်း Zero State နှင့် Zero Input Response တို့ရှိပါသည်။
- System တစ်ခုရဲ့ Zero State ၏ Frequency Response ကို H(s) ဟု ယူဆပါက H(s) ကို Transfer Function ဟုခေါ်သည်။

$$\mathbf{Y(s) = X(s)H(s)}$$

- X(s) Frequency Spectrum များရှိသော Input Signal တစ်ခု၏ Output Frequency Spectrum သည် Input Frequency Spectrum X(s) နှင့် Transfer Function H(s) တို့၏ မြှောက်လဒ် ဖြစ်သည်။
- Time Domain တွင်မူ Input Signal နှင့် Impulse Response တို့၏ Convolution သည် Output Signal ဖြစ်သည်။ ဒါဆိုရင် Transfer Function သည် Time Domain ရှိ Impulse Response ၏ Frequency Response ဖြစ်သည်။

$$\mathbf{y(t) = x(t) * h(t)}$$

Transfer Function

$$H(s) = \frac{Y(s)}{X(s)} \qquad \text{where } s = \sigma + j\omega$$

$$\overbrace{(b_0 D^0 + b_1 D^1 + b_2 D^2 + b_3 D^3 + \dots + b_n D^n)}^{Q(D)} y(t) = x(t) \overbrace{(a_0 D^0 + a_1 D^1 + a_2 D^2 + a_3 D^3 + \dots + a_n D^n)}^{P(D)}$$

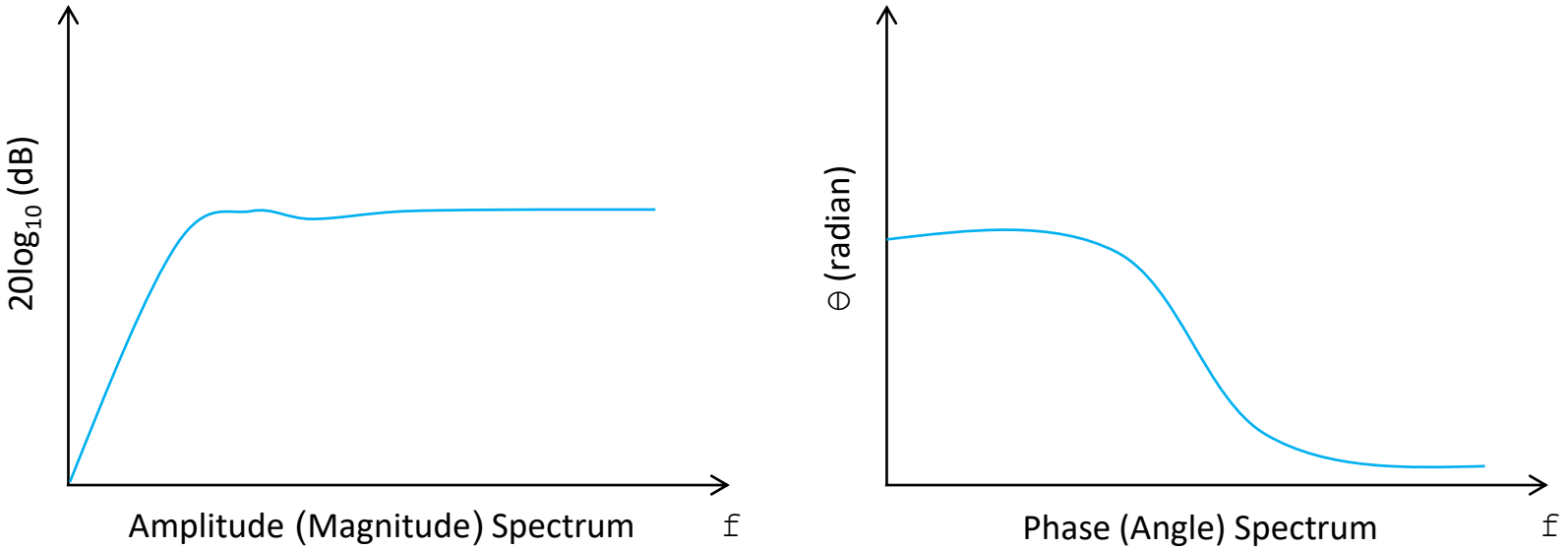
$$H(s) = \frac{Q(s)}{P(s)} = \frac{b_0 s^0 + b_1 s^1 + b_2 s^2 + b_3 s^3 + \dots + b_n s^n}{a_0 s^0 + a_1 s^1 + a_2 s^2 + a_3 s^3 + \dots + a_n s^n}$$

Zeroes

Poles

- Transfer Function ကို အများအားဖြင့် Input Frequencies (s_n) များနှင့် Output Frequencies (s_n) များ၏ Ratio အဖြစ် ဖော်ပြပါသည်။
- Numerator ရှိ (s_n) များကို Zeros ဟုခေါ်ပါသည်။ ဘာလို့လဲ ဆိုရင် Numerator ရှိ (s_n) များသာ Zero သာဆိုပါက Transfer Function သည် Zero ဖြစ်သွားမည် ဖြစ်သည်။ Denominator ရှိ s_n များကို Poles ဟုခေါ်ပါသည်။ ဘာလို့လဲ ဆိုရင် Denominator ရှိ s_n များသာ Zero သာဆိုပါက Transfer Function သည် Undefined (Infinity) ဖြစ်သွားမည် ဖြစ်သည်။
- Zeros နှင့် Poles များသည် Transfer Function ကို Solve လုပ်ရုံသာမက System Stability ကိုပါပြောပါသည်။

Bode Plot



- Frequency Response များကို Amplitude နှင့် Phase Spectrum များ ဖြင့်လေ့လာခြင်းကို Bode Plot များဖြင့် ဖော်ပြကြပါသည်။
- အမှန်တော့ H(s) Transfer Function ကို တခါတလေ Frequency Filter များအနေဖြင့် လည်းယူဆကြပါသည်။
- Low Pass Filter, High Pass Filter, Band Stop Filter နှင့် Band Pass Filter များသည် အင်မတန် အသုံးဝင်သော Filter များဖြစ်ကြသည်။ ဥပမာ၊ Music Player များတွင်ပါသော Equalizer များသည် Band Pass Filter များဖြစ်ကြသည်။

Frequency Domain Solutions of State Space

$$\mathbf{q}' = A\mathbf{q} + B\mathbf{u}$$

$$\mathbf{y} = C\mathbf{q} + D\mathbf{u}$$

$$Q(s) = \mathcal{L}(q)$$
$$U(s) = \mathcal{L}(u)$$
$$Y(s) = \mathcal{L}(y)$$

$$sQ(s) - q(0) = AQ(s) + BU(s)$$

$$sQ(s) - AQ(s) = q(0) + BU(s)$$

$$Q(s)(Is - A) = q(0) + BU(s)$$

$$Q(s) = (Is - A)^{-1}[q(0) + BU(s)]$$

$$\boldsymbol{\Phi}(s) = (Is - A)^{-1}$$

$$Q(s) = \boldsymbol{\Phi}(s)q(0) + \boldsymbol{\Phi}(s)BU(s)$$

$$q = \mathcal{L}^{-1}[\boldsymbol{\Phi}(s)q(0)] + \mathcal{L}^{-1}[\boldsymbol{\Phi}(s)BU(s)]$$

$$Y(s) = CQ(s) + DU(s)$$

$$Y(s) = C[\boldsymbol{\Phi}(s)q(0) + \boldsymbol{\Phi}(s)BU(s)] + DU(s)$$

$$Y(s) = C\boldsymbol{\Phi}(s)q(0) + [C\boldsymbol{\Phi}(s)B + D]U(s)$$

$$H(s) = C\boldsymbol{\Phi}(s)B + D$$

$$Y(s) = C\boldsymbol{\Phi}(s)q(0) + H(s)U(s)$$

$$y = \mathcal{L}^{-1}[C\boldsymbol{\Phi}(s)q(0)] + \mathcal{L}^{-1}[H(s)U(s)]$$

Time Domain Solutions of State Space

$$\mathbf{q}' = A\mathbf{q} + B\mathbf{u}$$

$$\mathbf{y} = C\mathbf{q} + D\mathbf{u}$$

$$\mathbf{q}(t) = e^{At}\mathbf{q}(0) + e^{At} * B\mathbf{u}(t)$$

$$\boldsymbol{\Phi}(t) = e^{At}$$

$$\mathbf{h}(t) = [C\boldsymbol{\Phi}(t)B + D\boldsymbol{\delta}(t)]$$

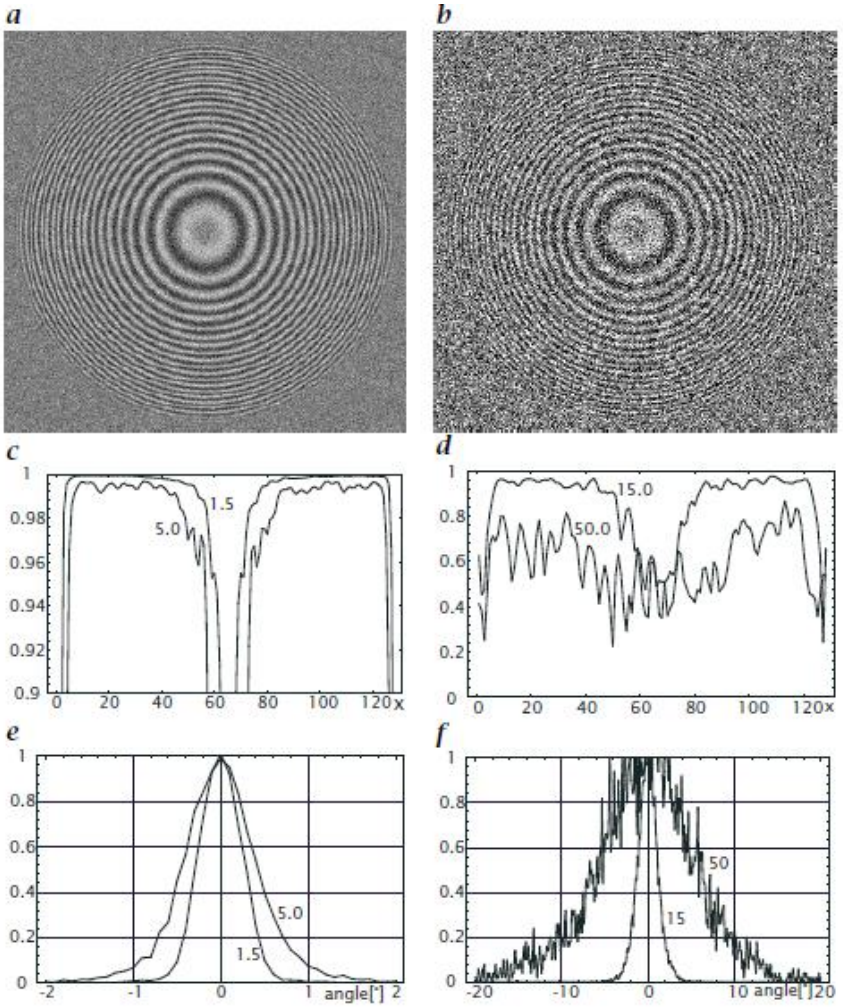
$$\mathbf{y}(t) = C\boldsymbol{\Phi}(t)\mathbf{q}(0) + \mathbf{h}(t) * \mathbf{u}(t)$$

$$\boldsymbol{L}[\mathbf{h}(t)] = H(s)$$

$$\boldsymbol{L}^{-1}[H(s)] = \mathbf{h}(t)$$

Applications of Fourier Transform

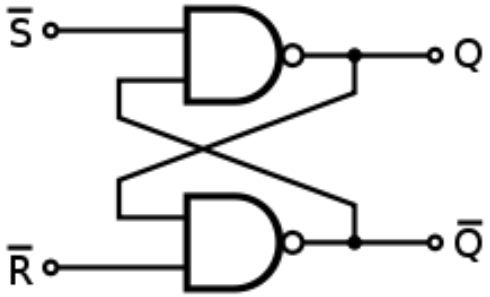
- ကျွန်တော်တို့ လက်တွေ့မှာလည်း Frequency Domain Analysis ကို အင်မတန်ကို သုံးကြပါသည်။
- Time Domain မှ Frequency Domain သို့ ပြောင်းရာတွင် အသုံးများဆုံးသော Algorithm များက FFT (Fast Fourier Transform) နှင့် DFT (Discrete Fourier Transform) တို့ ဖြစ်ပါသည်။
- အခုက သီအိုရီများကို ဦးစားပေးသည့် အတွက် Applications များကို သိပ်မပြောသေးပါ။ စိတ်ဝင်းစားရင်တော့ ကိုယ့်ဟာကိုယ် ဖတ်ကြည့်ထားလို့ ရပါသည်။
- အမှန်တော့ Image များကိုလည်း Signal အနေဖြင့် ယူဆလို့ ရပါသည်။ ဒါကြောင့် Image များကိုလည်း Fourier Transform လုပ်လို့ ရပါသည်။



Summary

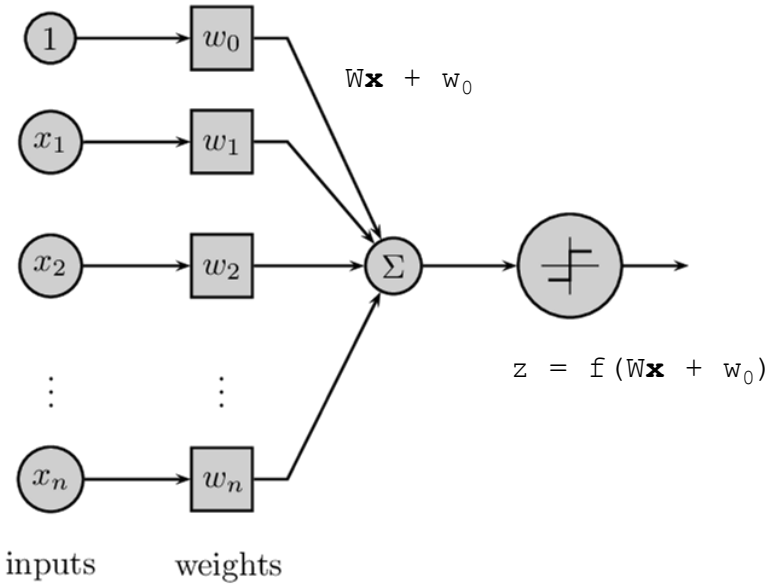
- အခုဆိုရင် ကျွန်တော်တို့ Signal Processing အပိုင်း နဲနဲလောက် သိသလောက်ရှိသွား ပါပြီ။
- အဓိကကတော့ ကျွန်တော်တို့ Signal များကို Linear Time Invariant Dynamical System များဖြင့် ဘယ်လို Process လုပ်သည်ကို Time Domain နှင့် Frequency Domain ရှုထောင့်များမှ လေ့လာခြင်း ဖြစ်သည်။
- အခုထိတော့ Theory တွေများပါသည်။ လက်တွေ့ပိုင်းတော့ ကျွန်တော်တို့ Signal Processing Lab ကျမှ အသေးစိတ်သွားပါမည်။
- နောက်တစ်ချက် အရေးကြီးသည်က Linear System များကို Linear Component များ၏ Linear Combination ဖြင့်တည်ဆောက်ထားခြင်း ဖြစ်သည်။ ဒီအချက်သည် အင်မတန် အရေးကြီးပါသည်။
- ဘာဖြစ်လို့လဲ ဆိုရင် Non-Linear Component များပါဝင်သော Non-Linear System များကို Machine Learning မှာ သုံးလို့ ဖြစ်ပါသည်။ Neuron (Perceptron) တစ်ခုသည် Linear Component တစ်ခု မဟုတ်သလို Neural Network ကြီးသည်လည်း Linear Network မဟုတ်ပါ။
- ဒီအပိုင်းကိုတော့ ကျွန်တော်တို့ နောက်မှ ဆွေးနွေးကြပါမည်။

Appendix A – Flip-flop



- ကျွန်တော်တို့ Linear Dynamical Component အကြောင်းကို ပြောခဲ့ပါသည်။
- Linear Dynamical Component များသည် Output = Input + Previous State ဟူသောအချက်ကို လိုက်နာပါသည်။
- Flip-Flop (Latch) များသည် ဒီ Concept (Dynamical State) ရဲ့ Digital Version တစ်ခု ဖြစ်ပါသည်။ အမှန်တော့ Flip-Flop သည် Non-Linear Component ဖြစ်သော်လည်း Digital Signal (State) များအတွက် ဒါသည်က ပြဿနာ မဟုတ်တော့ပါ။ Digital Signal များကိုယ်တိုင်က Non-Linear ဖြစ်၍ ဖြစ်သည်။
- အမှန်တော့ Latch များဖြင့် Computer များ၊ Digital Devices များ တည်ဆောက်ကြခြင်း ဖြစ်သည်။ တစ်နည်းအား Sequential Circuit များ၏ အသက်သည် Latch များဖြစ်ပြီး ဒီ Sequential Circuit များကြောင့်သာ Computer သည် Dynamical State Machine တစ်ခု ဖြစ်လာခြင်း ဖြစ်သည်။
- Flip-Flop များအကြောင်းကို တော့ ကျွန်တော်တို့ အချိန်ရရင် Computer Architecture and Organization မှာ ပြောပါမည်။

Appendix B – Perceptron

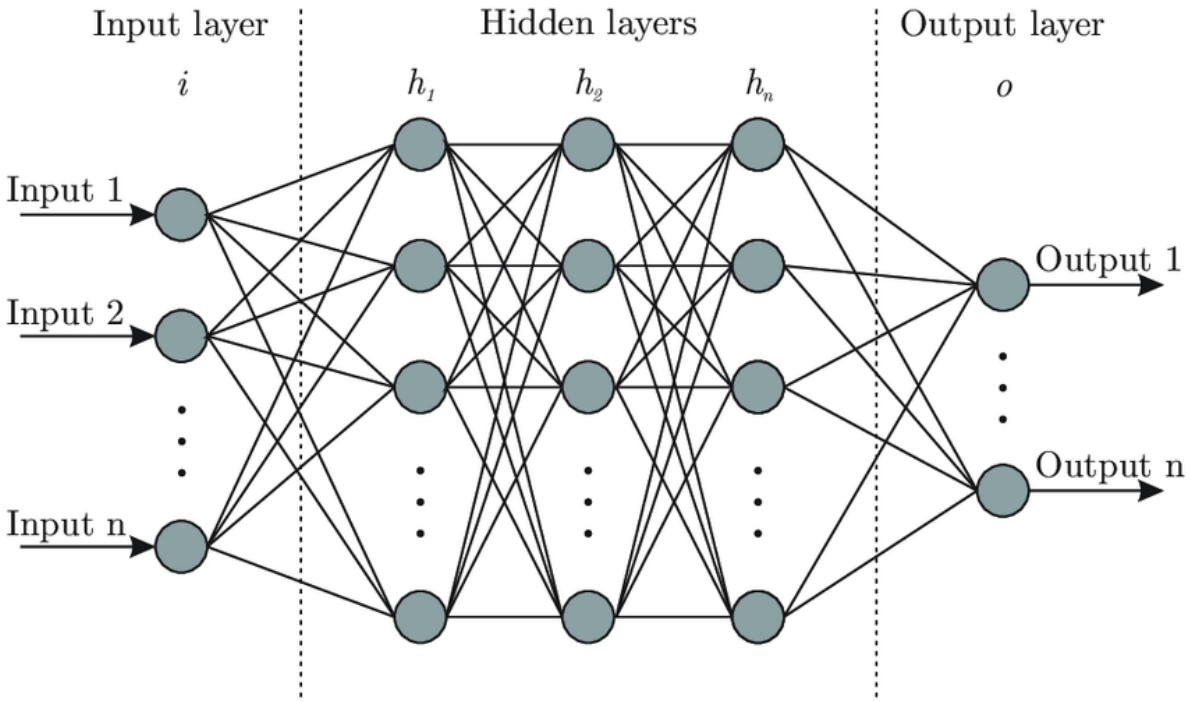


- အမှန်တော့ Perceptron တစ်ခုသည် Non-Linear Component တစ်ခု ဖြစ်သည်။ Perceptron တစ်ခုမှာ 2 ပိုင်းပါပါသည်။ ပထမပိုင်းက Weightများ နှင့် Input များ၏ Linear Combination ($\mathbf{W}\mathbf{x} + w_0$) ဖြစ်၍ Linear ဖြစ်သည်။ သို့သော် ဒုတိယပိုင်း Activation Function က Non-Linear ဖြစ်သည်။ Perceptron တစ်ခုကို Multi-Variable Scalar Function တစ်ခု အဖြစ် ယူဆနိုင်ပါသည်။

- ဒါဆိုရင် Perceptron မှာ State များကို ဘယ်မှာ Store လုပ်လဲလို့ မေးရင် Weights များမှာ လုပ်ပါသည်။ Weights များ၏ တန်ဖိုးသည် အရင်တန်ဖိုးများ ပေါ်မူတည်ပါသည်။ ထို့ကြောင့် Perceptron သည် State Dependent ဖြစ်သော Dynamical Component တစ်ခု ဖြစ်သည်။ သို့သော် Non-Linear ဖြစ်သည်။

- အမှန်တော့ Perceptron များပေါ်သည်က ကြာပြီ ဖြစ်သည်။ သို့သော် ကျွန်တော်တို့သည် Non-Linear Dynamical Component များကို ကောင်းကောင်း handle မလုပ်နိုင်ခဲ့သဖြင့် သိပ်ခေတ်မစားခဲ့တာ ဖြစ်သည်။ အခုတော့ တော်တော်လေး အသုံးများလာပြီ ဖြစ်သည်။

Appendix C – Neural Network



- Neural Network တစ်ခုသည် Neuron (Perceptron) များ၏ Network Model ဖြစ်သည်။
- Perceptron သည် Non-Linear Component တစ်ခု ဖြစ်သည့် အတွက် Neural Network တစ်ခုသည် Non-Linear Network တစ်ခု ဖြစ်သည်။