$$\overline{ReX}[k] = \frac{ReX[k]}{N/2}$$

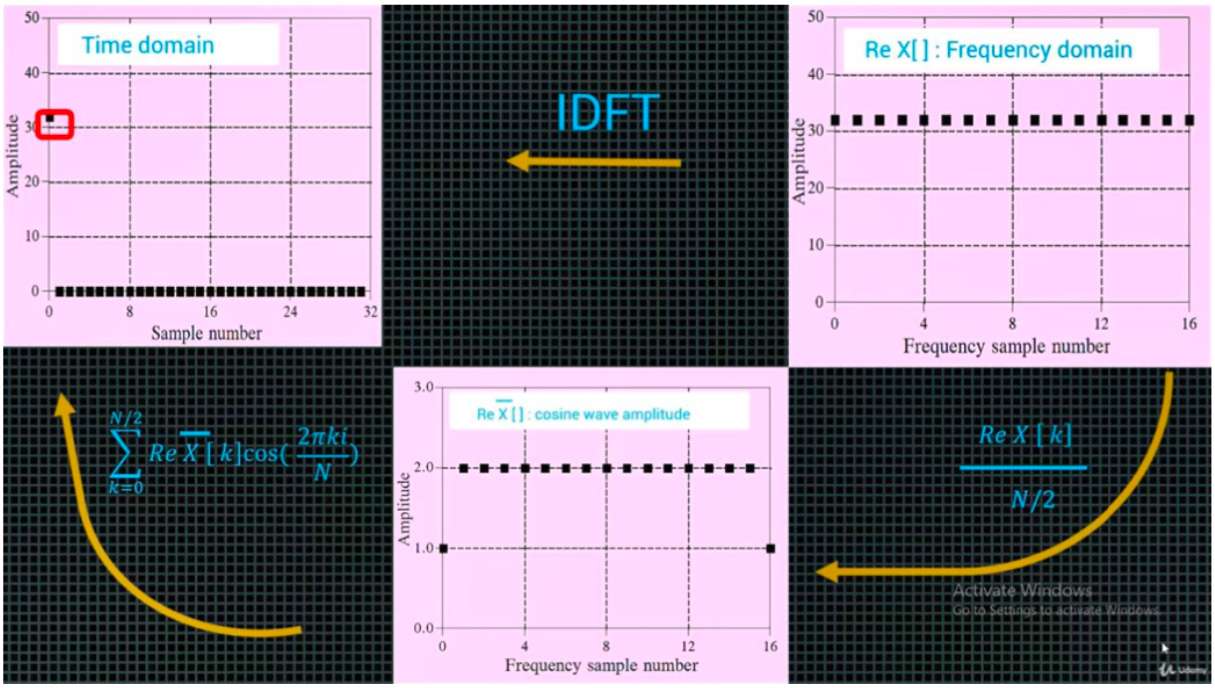
$$\overline{ImX}[k] = -\frac{ImX[k]}{N/2}$$

except for two special cases:

$$\overline{ReX}[0] = \frac{ReX[0]}{N}$$

$$\overline{ReX}[N/2] = \frac{ReX[N/2]}{N}$$

ပုံ(ခ) မှာပါတဲ့ frequency domain signal ကို ပုံ(ဂ) မှာပါတဲ့ cos wave တွေရဲ့ amplitude တွေအဖြစ် ပြောင်းလဲ ဖို့ရန်အတွက်



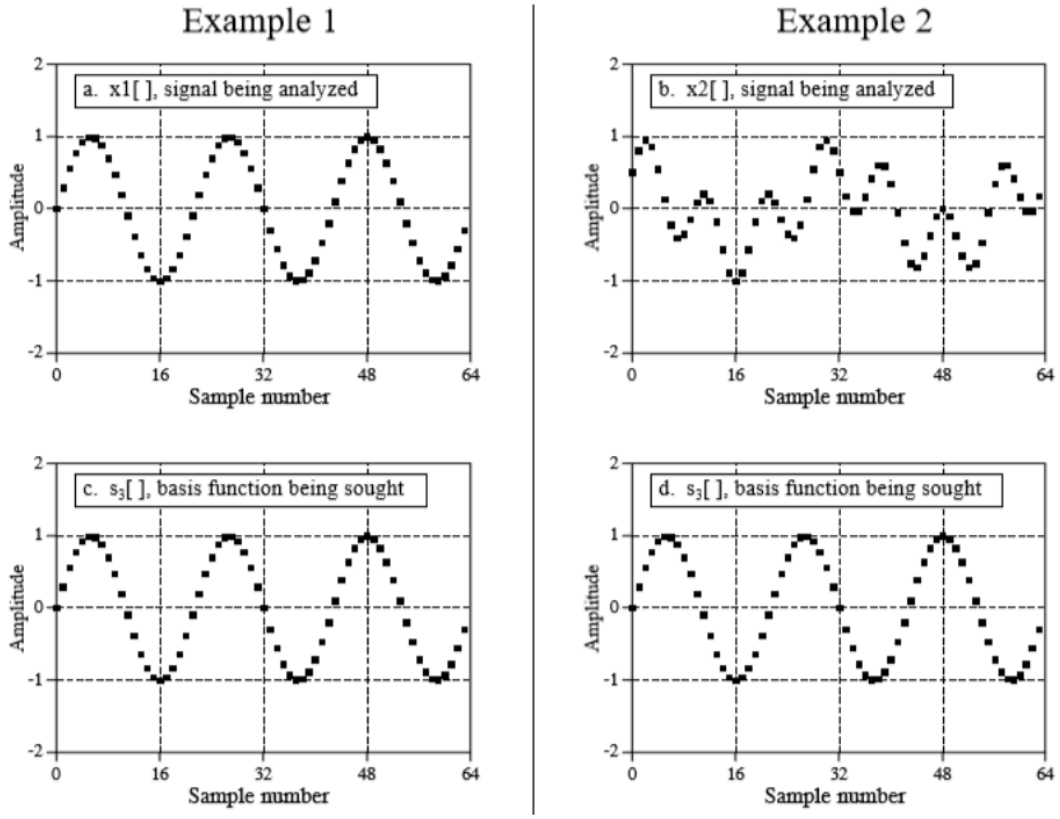
Calculating the Discrete Fourier Transform (DFT)

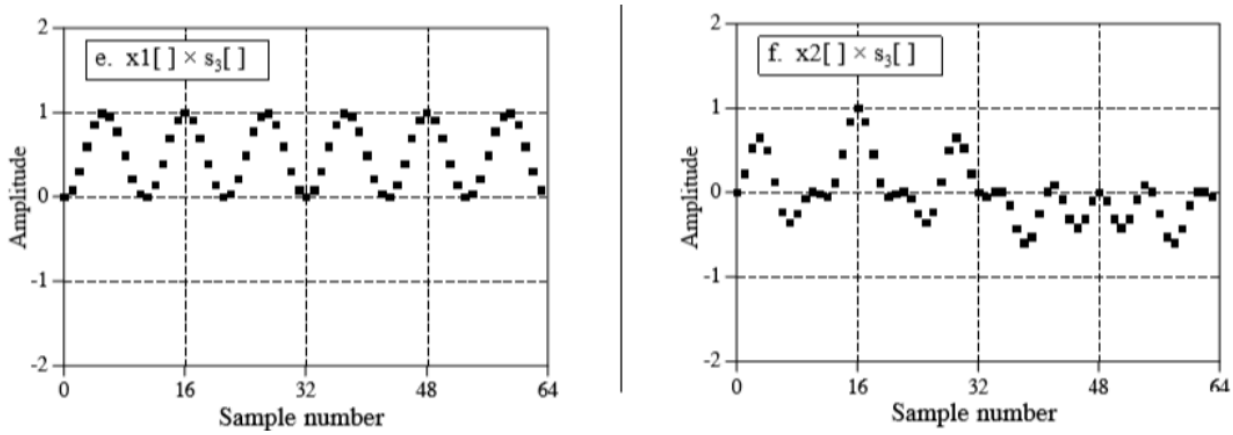
DFT ကိုနည်းလမ်းသုံးခုနဲ့တွက်လို့ရပါတယ်။

- Simultaneous Equations

- Correlation
- FFT

အဲဒီနည်းလမ်းသုံးခုထဲမှာ အခုဆွေးနွေးပြီးတော့ ကုဒ်ရေးမယ့် နည်းလမ်းကတော့ Correlation နည်းလမ်းဖြစ်ပါတယ်။ Sample Point ၆၄ ခုရှိတဲ့ signal တစ်ခုရဲ့ DFT ကိုတွက်ချက်မယ်ဆိုကြပါစို့။ N/2+1 အရေအတွက်ရင် Frequency domain ရဲ့ real part အတွက် 33 ပွိုင့်ရှိပြီး Frequency domain ရဲ့ imaginary part အတွက် 33 ပွိုင့်ရှိပါတယ်။ ဒီဥပမာလေးမှာတော့ - Im X[3] ဆိုတဲ့ sample ပွိုင့်တစ်ခုတည်းအတွက်ဘဲ တွက်ပြသွားမှာဖြစ်ပါတယ်။ (ImX[3] အတွက်အတွက်ဘဲတွက်မယ်ဆိုတော့ ဆိုလိုတာက - sample ပွိုင့် 0 ကနေ 63 ကြားထဲမှာ cycle သုံးခုစာ ထွက်မယ့် sine wave ရဲ့ amplitude ကိုတွက်ချက်ပြမှာဖြစ်ပါတယ်။) ဒီတစ်ခုတည်းကို ကြည့်ပြီးတော့ အခြားသော frequency domain တွေအတွက်လည်း တွက်ချက်နိုင်မှာဖြစ်ပါတယ်။





Correlation နည်းလမ်းကိုအသုံးပြုပြီးတော့ Im X[3] ကိုတွက်ထားတဲ့ ပုံကို အောက်ပါပုံမှာပြထားပါတယ်။ ပုံ(က) နဲ့ ပုံ(ခ) မှာပြထားတာက time domain signal နှစ်ခုဖြစ်တဲ့ x1[] နဲ့ x2[] ဖြစ်ပါတယ်။ ပထမ signal တစ်ခုဖြစ်တဲ့ x1[] မှာ sample ပွိုင့် 0 နဲ့ 63 ကြား cycle သုံးခုစာထွက်စေတဲ့ sine wave ရှိပါတယ်။

အခြား signal တစ်ခုမှာပါဝင်တဲ့ waveform ကိုသိချင်တယ်ဆိုရင် - signal နှစ်ခုကို မြှောက်ပြီးတော့ ရလာမယ့် signal ထဲမှာ sample ပွိုင့်တွေကိုပေါင်းပေးရမှာဖြစ်ပါတယ်။ ပုံ(ဂ) နဲ့ ပုံ(ဃ) နှစ်ခုစလုံး မှာ ကျွန်တော်တို့လိုချင် တဲ့ ကျွန်တော်တို့ရှာနေတဲ့ sample နံပါတ် 0 နဲ့ 63 ကြားထဲမှာရှိသော cycle သုံးခုစာ output ထွက်စေတဲ့ sine wave ကို ပြလိုက်ပါတယ်။ ပုံ(င) ကတော့ ပုံ(က) နဲ့ ပုံ(ဂ) ကိုမြှောက်လို့ရလာတဲ့ရလဒ်ဖြစ်ပါတယ်။ ပုံ (စ) ကတော့ ပုံ(ခ) နဲ့ ပုံ(ဃ) ကိုမြှောက်လို့ရလာတဲ့ရလဒ်ဖြစ်ပါတယ်။

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

$$ImX[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi k i / N)$$

Frequency domain ထဲမှာရှိတဲ့ အခြားသော sample တွေအတွက်တွက်တဲ့အခါမှာလည်း အထက်ပါ အဓိကွေးရှင်းကိုသုံးပြီးတော့ တွက်လို့ရပါတယ်။ ဒီအဓိကွေးရှင်းမှာ x[i] က time domain signal ဖြစ်ပြီးတော့ ReX[k] နဲ့ ImX[k] က frequency domain signal တွေဖြစ်ပါတယ်။ I တန်ဖိုးက သုညကနေ N-1 အထိ အလုပ်လုပ်ပြီး k တန်ဖိုး ကတော့ 0 ကနေ N/2 အထိအလုပ်လုပ်ပါတယ်။

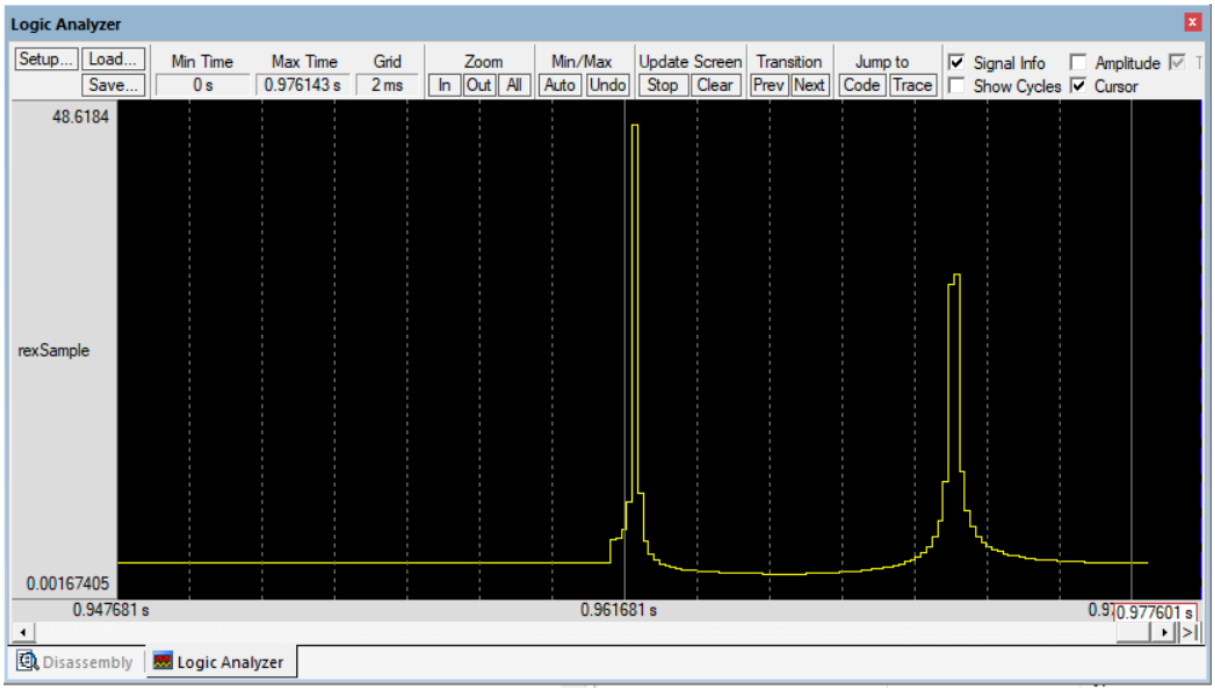
Frequency domain ထဲမှာရှိတဲ့ sample တစ်ခုချင်းစီဟာ time domain signal ကို မိမိလိုချင်တဲ့ sine wave သို့မဟုတ် cosine wave နဲ့မြှောက်ပြီးတော့ ရလာတဲ့ point တွေကိုပေါင်းတာနဲ့ တူတူပါဘဲ။

STM32F411RE မှာ DFT အတွက် program လေးရေးကြည့်လိုက်ရအောင်။

```
#include "stm32f4xx_hal.h"            // Keil::Device:STM32Cube HAL:Common
        #include "arm_math.h"                 // ARM::CMSIS:DSP
        #define       SIG_LENGTH        200
        #define IMP_RSP_LENGTH        29
        extern void SystemClock_Config(void);
        extern float32_t inputSignal_f32_1kHz_15kHz[SIG_LENGTH];

        float32_t REX[SIG_LENGTH/2 ];

        float32_t IMX[SIG_LENGTH/2];

    void plot_input_signal(void);       void plot_both_signal(void);

    float32_t inputSample;
    float32_t rexSample;
    float32_t imxSample;

    void plot_imp_response(void);
    void plot_output_signal(void);
    void plot_all(void);
    void calc_sig_dft(float32_t *sig_src_arr, float32_t *sig_dest_rex_arr, float32_t
*sig_dest_imx_arr, uint32_t sig_length);
    void plot_rex_signal(void);
    void get_dft_output_mag(void);

    uint32_t freq;

    int main()    {
        HAL_Init();
        SystemClock_Config();

        calc_sig_dft(&inputSignal_f32_1kHz_15kHz[0],&REX[0], &IMX[0],SIG_LENGTH);

        get_dft_output_mag();
```

```
  plot_rex_signal();

        while(1) {

        }

}

 void calc_running_sum(float32_t *sig_src_arr,float32_t *sig_dest_arr,uint32_t sig_length)
{
         int i;
        sig_src_arr[0] = sig_dest_arr[0];
         for(i=0;i<sig_length;i++)  {
                 sig_dest_arr[i]= sig_dest_arr[i-1]+sig_src_arr[i];
         }
}
void calc_first_difference(float32_t *sig_src_arr,float32_t *sig_dest_arr, uint32_t
sig_length) {

        sig_dest_arr[0] =0;
        int  i;
        for(i=0;i<sig_length;i++) {
                sig_dest_arr[i] = sig_src_arr[i] - sig_src_arr[i-1];
        }
}

void calc_sig_dft(float32_t *sig_src_arr, float32_t *sig_dest_rex_arr, float32_t
*sig_dest_imx_arr, uint32_t sig_length) {
        int i,k,j;
        for(j=0;j<(sig_length/2);j++)      {
                sig_dest_rex_arr[j] =0;
                 sig_dest_imx_arr[j] =0;
        }
```

```
        for(k=0;k<(sig_length/2);k++) {
                for(i=0;i<sig_length;i++) {
sig_dest_rex_arr[k] =  sig_dest_rex_arr[k] + sig_src_arr[i]*cos(2*PI*k*i/sig_length);
 sig_dest_imx_arr[k] =  sig_dest_imx_arr[k] - sig_src_arr[i]*sin(2*PI*k*i/sig_length);
                }
        }
}
        void get_dft_output_mag(void)    {
                int k;
                for(k=0;k<(SIG_LENGTH/2);k++){
                        REX[k] = fabs(REX[k]);
                }
        }
        void plot_rex_signal(void)  {
                int i,j;
                for(i=0;i<(SIG_LENGTH/2);i++)  {
                rexSample = REX[i];
                for(j=0;j<3000;j++){}
        }
}
void SysTick_Handler(void) {
        HAL_IncTick();
        HAL_SYSTICK_IRQHandler();
}
```



void calc_sig_dft(float32_t ***sig_src_arr**, float32_t ***sig_dest_rex_arr**, float32_t ***sig_dest_imx_arr**, uint32_t **sig_length**)

ဒီ ဖန်ရှင်မှာ ပထမ argument ဖြစ်တဲ့ **sig_src_arr** ကတော့ **time domain signal** ဖြစ်ပါတယ်။ **sig_dest_rex_arr** ကတော့ frequency domain ရဲ့ **real part** ဖြစ်ပြီးတော့ **sig_dest_imx_arr** ကတော့ frequency domain ရဲ့ imaginary part ဖြစ်ပါတယ်။ **sig_length** ကတော့ signal ရဲ့ length ဖြစ်ပါတယ်။

အဲဒီ ဖန်ရှင်ထဲမှာဘဲ output signal နှစ်ဖြစ်တဲ့ **sig_src_arr** နဲ့ **sig_dest_imx_arr** နှစ်ခုကို initialize အနေနဲ့ 0 ပေးခဲ့လိုက်ဖို့ရန်အတွက် အောက်ပါအတိုင်း ရေးခဲ့ပါတယ်။

```
for(j=0;j<(sig_length/2);j++){
        sig_dest_rex_arr[j] =0;
        sig_dest_imx_arr[j] =0;
}
```

ဒီနေရာမှာဘာကြောင့် **sig_length/2** နဲ့စစ်ထားသလဲဆိုတော့ DFT Calculation ပြီးသွားအခါ signal ရဲ့ **real part** က time domain signal ရဲ့တစ်ဂက်ဖြစ်နေပြီး၊ real part ရဲ့ length ကလည်း time domain signal ရဲ့တစ်ဂက်ဖြစ်နေပါတယ်။ ထို့အတူ imaginary part ရဲ့ length ကလည်း time domain signal ရဲ့တစ်ဂက်ဖြစ်

နေပါတယ်။ ဒါကြောင့်မို့လို့ for loop ထဲမှာ array နှစ်ခုစလုံးရဲ့ length ကို sig_length ရဲ့ တစ်ဝက်ဖြစ်အောင် လုပ်ထားတာဖြစ်ပါတယ်။

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

$$ImX[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi k i / N)$$

အထက်မှာပြောခဲ့တဲ့ အီကွေးရှင်းအရအောက်ပါကုဒ်ကိုရေးပါတယ်။

```
for(k=0;k<(sig_length/2);k++)  {
        for(i=0;i<sig_length;i++)  {

sig_dest_rex_arr[k] =  sig_dest_rex_arr[k] + sig_src_arr[i]*cos(2*PI*k*i/sig_length);

 sig_dest_imx_arr[k] =  sig_dest_imx_arr[k] - sig_src_arr[i]*sin(2*PI*k*i/sig_length);

        }

}
```

ECG Signal

ECG ဆိုတာ Electrocadiography ဖြစ်ပါတယ်။ လူရဲ့ အရေပြားအပေါ်မှာ electrode တွေသုံးပြီးတော့ လူ့နှလုံးရဲ့ activity ကို graph တစ်ခုအနေနဲ့ပြပေးတာဖြစ်ပါတယ်။

MatLab ကနေ ထုတ်ထားသော 640 points ရှိတဲ့ ECG Signal တစ်ခုကို waveform.c မှာ အောက်ပါအတိုင်း ထပ်ထည့်လိုက်ပါမယ်။

```
float32_t _640_points_ecg_[640]=
{0,0.0010593,0.0021186,0.003178,0.0042373,0.0052966,0.0063559,0.0074153,0.0084746,0.045198,0.081921,0.11864,0.15537,0.19209,0.22881,0.26554,0.30226,0.33898,0.30226,0.26554,0.22881,0.19209,0.15537,0.11864,0.081921,0.045198,0.0084746,0.0077684,0.0070621,0.0063559,0.0056497,0.0049435,0.0042373,0.0035311,0.0028249,0.0021186,0.0014124,0.00070621,0,-0.096045,-0.19209,-0.28814,-0.073446,0.14124,0.35593,0.57062,0.78531,1,0.73729,0.47458,0.21186,-0.050847,-0.31356,-0.57627,-0.83898,-0.55932,-0.27966,0,0.00073692,0.0014738,0.0022108,0.0029477,0.0036846,0.0044215,
0.0051584,0.0058954,0.0066323,0.0073692,0.0081061,0.008843,0.00958,0.010317,0.011054,0.011791,0.012528,0.013265,0.014001,0.014738,0.015475,0.016212,0.016949,0.03484,0.052731,0.070621,0.088512,0.1064,0.12429,0.14218,0.16008,0.17797,0.16186,0.14576,0.12966,0.11356,0.097458,0.081356,0.065254,0.049153,0.033051,0.016949,0.013559,0.010169,0.0067797,0.003389
```