Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – hoseong Lee(이호성)

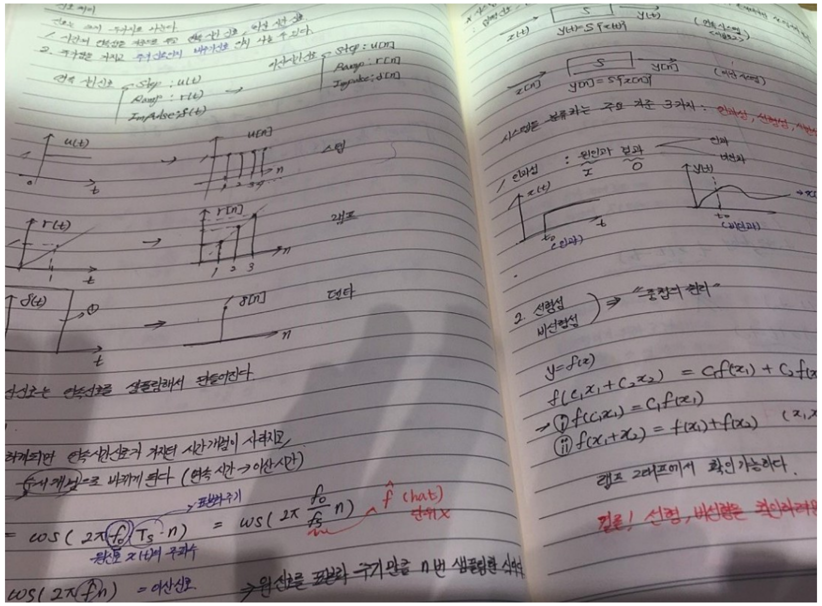
hslee00001@naver.com

목차. DSP , 신호처리 6/11

목차 control + 오른쪽 마우스 클릭

1. LTI 시스템

https://m.blog.naver.com/ldj1725/220426095029#
신호처리 아주 잘 설명되어 있음. 참고.



신호처리

신호는 크게 두가지로 나뉜다.

1. 시간에 연속성을 기준으로 [illegible] 연속시간신호, 이산시간신호
2. 주기성을 가지고 주기신호이냐 비주기신호이냐 나눌 수 있다.

연속시간신호 — Step : $u(t)$, Ramp : $r(t)$, Impulse : $\delta(t)$ → 이산시간신호 — Step : $u[n]$, Ramp : $r[n]$, Impulse : $\delta[n]$

$u(t)$
$u[n]$
스텝
$r(t)$
$r[n]$
램프
$\delta(t)$
$\delta[n]$
델타

[illegible]신호는 연속신호를 샘플링해서 만들어진다

[illegible]하게되면 연속시간신호가 가지던 시간개념이 사라지고, 순서개념으로 바뀌게 된다 (연속 시간 → 이산시간)

$= \cos(2\pi f_0 T_s \cdot n) = \cos(2\pi \frac{f_0}{f_s} n)$ → $\hat{f}$ (hat) 단위X

원신호 $x(t)$의 주파수 / 표본화주기

$\cos(2\pi \hat{f} n)$ = 이산신호. → 원신호를 표본화 주기만큼 n번 샘플링한 식이다

$x(t)$ → $\boxed{S}$ → $y(t)$, $y(t) = S\{x(t)\}$ (연속 시스템)

$x[n]$ → $\boxed{S}$ → $y[n]$, $y[n] = S\{x[n]\}$ (이산 시스템)

시스템을 분류하는 주요 기준 3가지 : 인과성, 선형성, 시불변성

1. 인과성 : 원인과 결과 (I, O) — 인과 / 비인과

2. 선형성 / 비선형성 ⇒ "중첩의 원리"

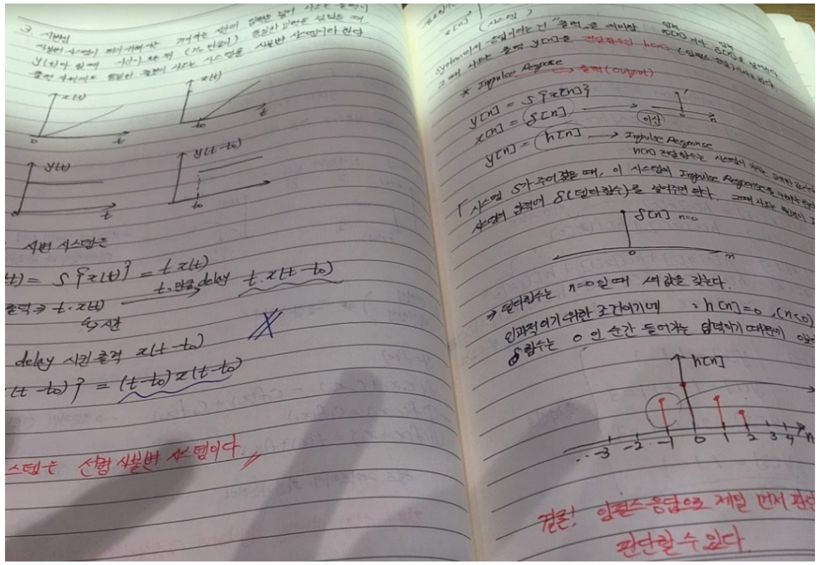
$y = f(x)$

$f(c_1x_1 + c_2x_2) = c_1f(x_1) + c_2f(x_2)$

→ (i) $f(c_1x_1) = c_1f(x_1)$
(ii) $f(x_1 + x_2) = f(x_1) + f(x_2)$

램프 그래프에서 확인 가능하다.

결론! 선형, 비선형은 중첩의 원리[illegible]



3. 시변성

[illegible]

$x(t)$
$x(t)$
t_0
$y(t)$
$y(t-t_0)$
t_0

시변 시스템은

$y(t) = S\{x(t)\} = t\,x(t)$ —t_0만큼 delay→ $t \cdot x(t-t_0)$

출력 ⇒ $t \cdot x(t)$ 시간

delay 시킨 출력 $x(t-t_0)$

$\{x(t-t_0)\} = (t-t_0)x(t-t_0)$

[illegible]스템은 선형 시불변 시스템이다

(시스템)

System에서 응답이라는 건 "출력"을 의미함. [illegible] 출력 $y[n]$을 전달함수인 $h[n]$ (임펄스 응답)[illegible]

Impulse Response ⟺ 출력(output)

* $y[n] = S\{x[n]\}$
$x[n] = \delta[n]$ (이산)
$y[n] = h[n]$ → Impulse Response

$h[n]$ 전달함수는 시스템이 [illegible]

시스템 S가 주어졌을 때, 이 시스템의 Impulse Response를 구하[illegible] 시스템의 입력에 δ(델타함수)를 넣어주면 된다. [illegible]

$\delta[n]$ n=0
0
n

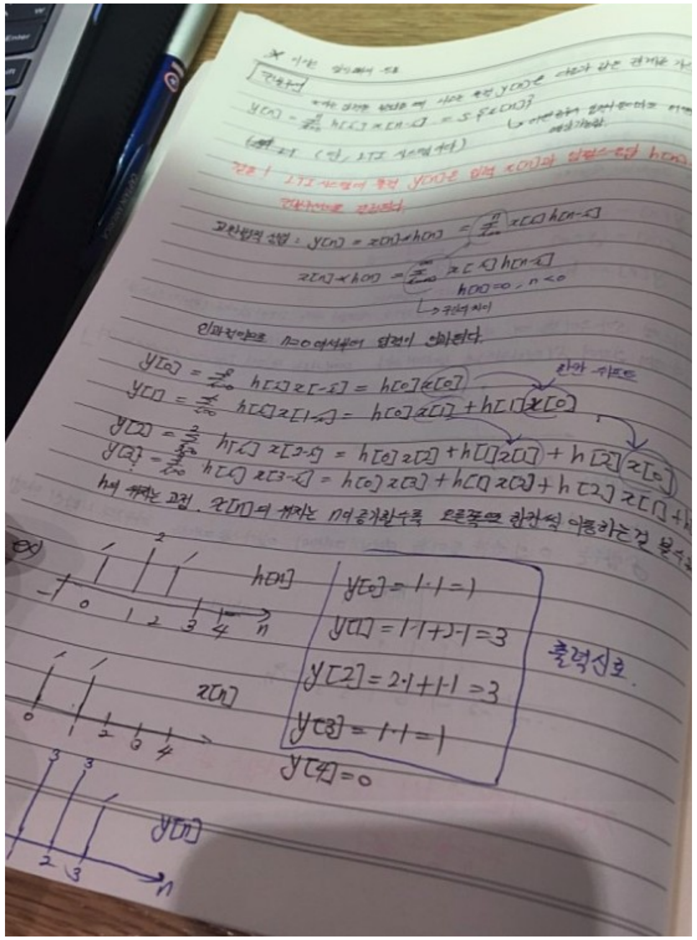
델타함수는 n=0일 때 1의 값을 갖는다.

⇒ 인과적이기 위한 조건이기에 $= h[n] = 0, (n<0)$

δ함수는 0인 순간 들어가는 입력이기 때문에 [illegible]

$h[n]$
-3
-2
-1
0
1
2
3
4
n

결론! 임펄스응답으로 제일 먼저 판[illegible] 판단할 수 있다.



※ 이산 [illegible]

컨볼루션

$y[n] = \sum h[k]x[n-k] = S\{x[n]\}$

[illegible] (단, LTI 시스템 [illegible])

결론! LTI 시스템의 출력 $y[n]$은 입력 $x[n]$과 임펄스응답 $h[n]$의 컨볼루션으로 결정된다

교환법칙 성립 : $y[n] = x[n] * h[n] = \sum_{k} x[k]h[n-k]$

$x[n] * h[n] = \sum_{k} x[k]h[n-k]$, $h[n] = 0,\ n<0$

인과적이므로 n=0에서부터 입력이 입력된다.

$y[0] = \sum_{k=0}^{0} h[k]x[-k] = h[0]x[0]$

$y[1] = \sum_{k=0}^{1} h[k]x[1-k] = h[0]x[1] + h[1]x[0]$ 한칸 시프트

$y[2] = \sum_{k=0}^{2} h[k]x[2-k] = h[0]x[2] + h[1]x[1] + h[2]x[0]$

$y[3] = \sum_{k=0}^{3} h[k]x[3-k] = h[0]x[3] + h[1]x[2] + h[2]x[1] + [illegible]$

h의 위치는 고정. x[n]의 위치는 n이 증가할수록 오른쪽으로 한칸씩 이동하는걸 볼 수 [illegible]

ex)

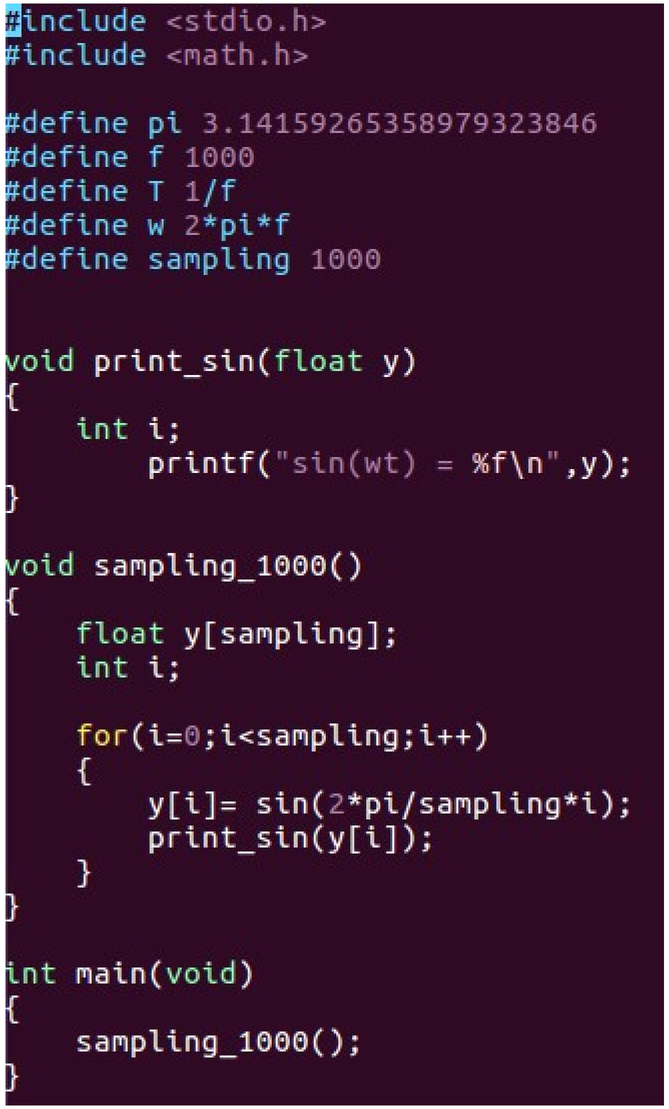
$y[0] = 1 \cdot 1 = 1$
$y[1] = 1 \cdot 1 + 2 \cdot 1 = 3$
$y[2] = 2 \cdot 1 + 1 \cdot 1 = 3$
$y[3] = 1 \cdot 1 = 1$
$y[4] = 0$

출력신호.

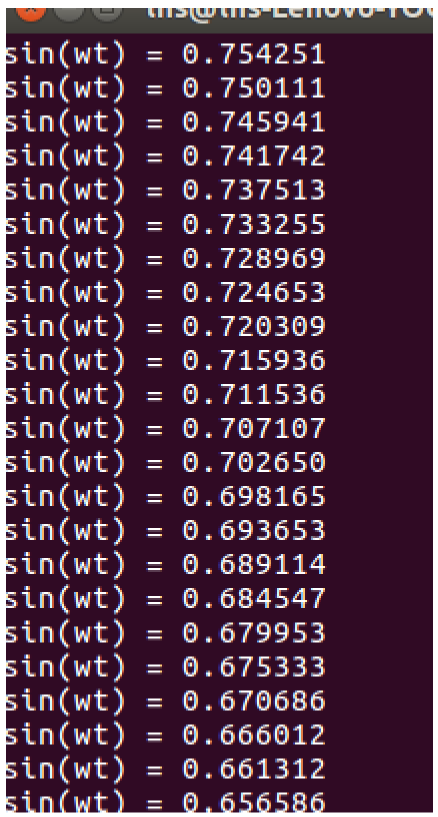
2. openGL 설치

```
-sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install freeglut3 freeglut3-dev
sudo apt-get install glew-utils glee-dev
```

3.1 sin(wt) 만들기 1



```c
#include <stdio.h>
#include <math.h>

#define pi 3.14159265358979323846
#define f 1000
#define T 1/f
#define w 2*pi*f
#define sampling 1000


void print_sin(float y)
{
    int i;
        printf("sin(wt) = %f\n",y);
}

void sampling_1000()
{
    float y[sampling];
    int i;

    for(i=0;i<sampling;i++)
    {
        y[i]= sin(2*pi/sampling*i);
        print_sin(y[i]);
    }
}

int main(void)
{
    sampling_1000();
}
```

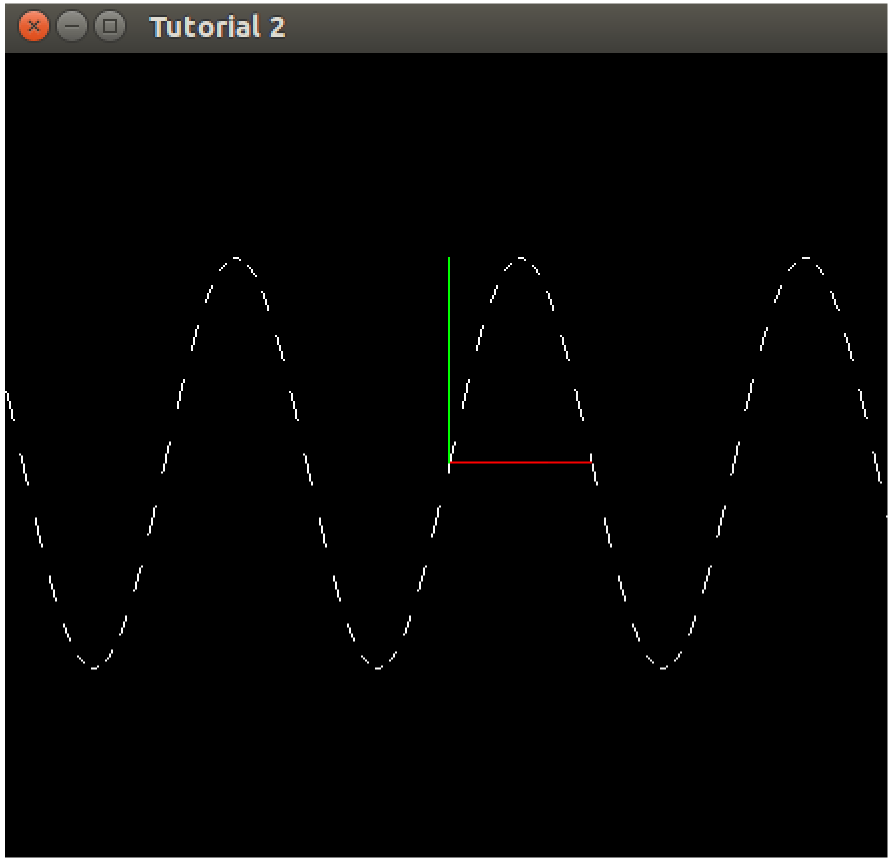


```
sin(wt) = 0.754251
sin(wt) = 0.750111
sin(wt) = 0.745941
sin(wt) = 0.741742
sin(wt) = 0.737513
sin(wt) = 0.733255
sin(wt) = 0.728969
sin(wt) = 0.724653
sin(wt) = 0.720309
sin(wt) = 0.715936
sin(wt) = 0.711536
sin(wt) = 0.707107
sin(wt) = 0.702650
sin(wt) = 0.698165
sin(wt) = 0.693653
sin(wt) = 0.689114
sin(wt) = 0.684547
sin(wt) = 0.679953
sin(wt) = 0.675333
sin(wt) = 0.670686
sin(wt) = 0.666012
sin(wt) = 0.661312
sin(wt) = 0.656586
```

… …

3.2 sin(wt) 2 – openGL 로 파형 보기

y = amplitude * sin(k * x);

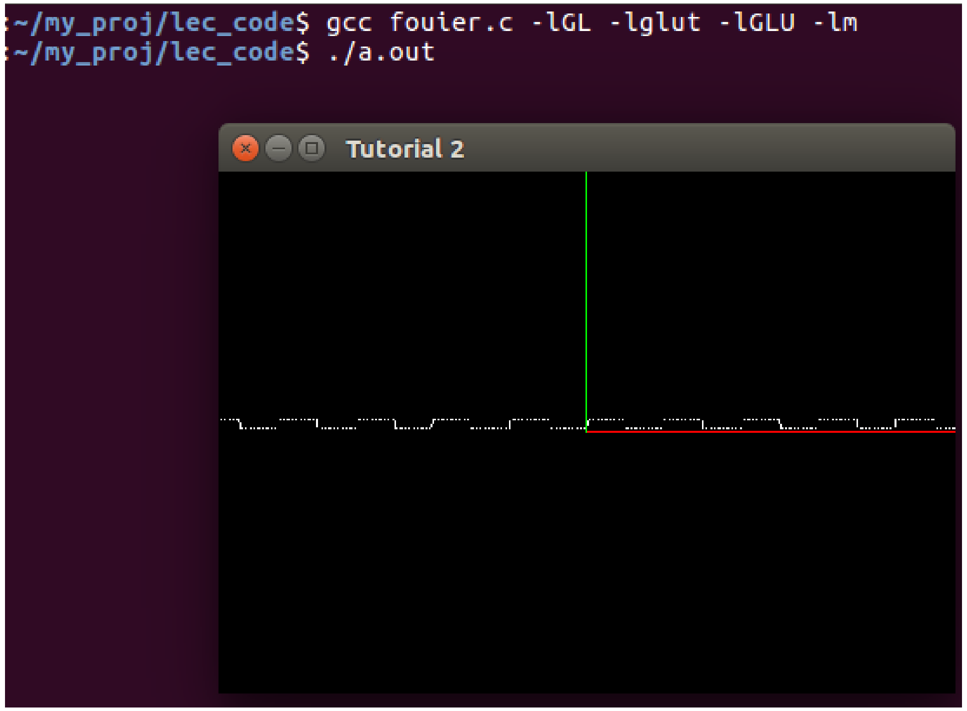
```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>
#define pi 3.14159265358979323846

void originAxis(void);
void sineWave(void);
void idle(void);

void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  originAxis();
  sineWave();

  //printf("%s\n", gluErrorString(glGetError()));
  glutSwapBuffers();
}

void sineWave(void)
{
  float wavelength = 2;
  float amplitude = 1;
  float inc = 0.05;
```
	결과
	

```
   float k, x, y;
   glBegin(GL_LINES);
   glColor3f(1,1,1);
  for(x=-10*pi;x<=10*pi;x+=inc){
     k = 2 * 3.14 / wavelength;
     y = amplitude * sin(k * x);
     glVertex3f(x, y, 0);
   }
   glEnd();

}
void idle(void)
{
   float wavelength = 2;   // 주기
   float amplitude = 1;    // 진폭
   float inc = 0.05;       // 증가폭 (샘플링 주기)
   float k, x, y; // k= 각주파수
   for (x = -1; x <= 1; x += inc){
     glBegin(GL_POINTS);
     glPointSize(200);
     glColor3f(0, 1, 0);
     k = 2 * 3.14 / wavelength;
     y = amplitude * sin(k * x);
     glVertex3f(x, y, 0);
     glEnd();
   }

   glutPostRedisplay();
}

void originAxis(void)
```

```
{
   glBegin(GL_LINES);
   glColor3f(1,0,0); // red
   glVertex3f(0,0,0);
   glVertex3f(1, 0, 0); // x
   glColor3f(0,1,0); // green
   glVertex3f(0,0,0);
   glVertex3f(0, 1, 0); // y
   glColor3f(0,0,1); // blue
   glVertex3f(0,0,0);
   glVertex3f(0, 0, 1); // z     --> 우린 지금 2차원으로 만들어놔서 그림에 표시가
>안됌.
   glEnd();
}

int main(int argc, char **argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH); // vga_rgb ,,depth -> 3D
   glutCreateWindow("Tutorial 2");   //제목

   glOrtho(-pi, pi, -2, 2, -1, 1);     // 직교좌표계
   glEnable(GL_DEPTH_TEST);          // 깊이값 설정해라.

   glutDisplayFunc(display);         // 화면에 뿌려라.
   glutIdleFunc(idle);
   glutMainLoop();

   return EXIT_SUCCESS;
}
```

4. 1 푸리에 급수로 사각파 만들기

y = y+ (1/(n*pi)*(1-cos(n*pi))*sin(n*x));

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>

#define pi 3.14159265358979323846
#define f 1000
void originAxis(void);
void sineWave(void);
void idle(void);

void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  originAxis();
  sineWave();
  //printf("%s\n", gluErrorString(glGetError()));
  glutSwapBuffers();
}

void sineWave(void)
{
  int n;
  float wavelength = 2;
  float amplitude = 1;
```
	
	```
	:~/my_proj/lec_code$ gcc fouier.c -lGL -lglut -lGLU -lm
	:~/my_proj/lec_code$ ./a.out
	```

```
  float inc = 0.05;
  float k, x, y;
  glBegin(GL_LINES);
  glColor3f(1,1,1);
  for(x=-10*pi;x<=10*pi;x+=inc){
        // k = 2 * pi / wavelength;
        // y = amplitude * sin(k * x);
    y=1;
    for(n=1;n<10000;n++){
      y = y+ (1/(n*pi)*(1-cos(n*pi))*sin(n*x));
    }
    y = y + 1/2;
    glVertex3f(x, y, 0);
  }
  glEnd();

}

void idle(void)
{
  int n;
  float wavelength = 2;   // 주기
  float amplitude = 1;    // 진폭
  float inc = 0.05;       // 증가폭 (샘플링 주기)
  float k, x, y; // k= 각주파수
  for (x = -1; x <= 1; x += inc){
    glBegin(GL_POINTS);
    glPointSize(200);
    glColor3f(0, 1, 0);
  k = 2 * 3.14 / wavelength;
  y = amplitude * sin(k * x);
    glVertex3f(x, y, 0);
    glEnd();
  }

  glutPostRedisplay();
}

void originAxis(void)
{
  glBegin(GL_LINES);
  glColor3f(1,0,0); // red
  glVertex3f(0,0,0);
  glVertex3f(30, 0, 0); // x
  glColor3f(0,1,0); // green
  glVertex3f(0,0,0);
  glVertex3f(0, 30, 0); // y
  glColor3f(0,0,1); // blue
  glVertex3f(0,0,0);
  glVertex3f(0, 0, 30); // z     --> 우린 지금 2차원으로 만들어놔서 그림에 표시
가 안됨.
  glEnd();
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH); // vga_rgb ,,depth -> 3D
  glutCreateWindow("Tutorial 2");   //제목
  glutInitWindowSize(1000,500);
  glOrtho(-30, 30, -30, 30, -30, 30);     // 직교좌표계
  glEnable(GL_DEPTH_TEST);          // 깊이값 설정해라.

  glutDisplayFunc(display);         // 화면에 뿌려라.
  glutIdleFunc(idle);
  glutMainLoop();

  return EXIT_SUCCESS;
}
```

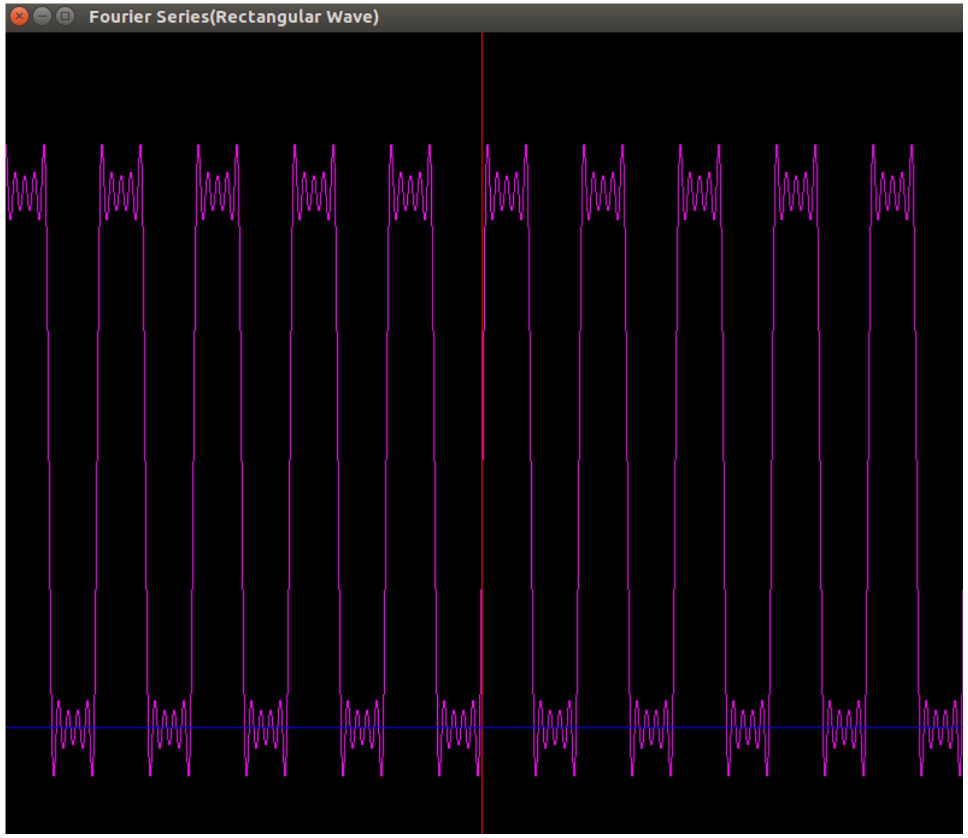
4.2 openGL 으로 사각파 출력

yp += ((1.0 - cos(i * M_PI)) / (i * M_PI)) * sin(i * x);

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>

void originAxis(void);
void sineWave(void);
void idle(void);

void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  originAxis();
  sineWave();

  glutSwapBuffers();
}

void sineWave(void)
{
  float wavelength = 2.0 * M_PI;
  float amplitude = 1;
  float inc = 2.0 * M_PI / 1024.0;
  float k, x, y, yp = 0, y2, y2p = 0, cx, cy, cy2;
```

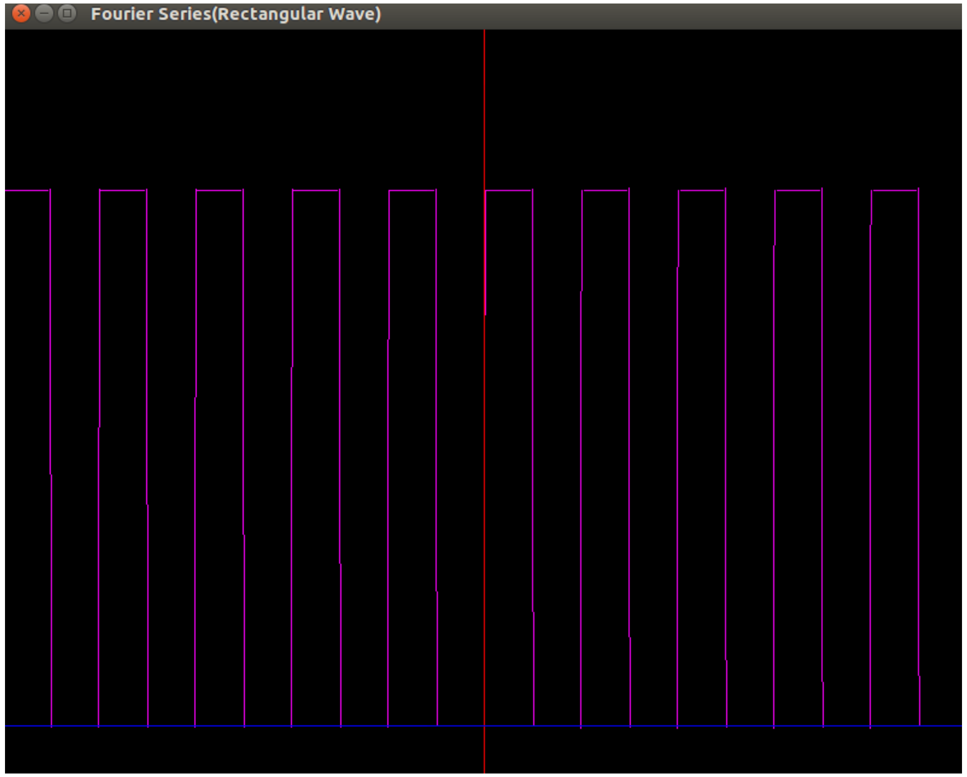
결과

푸리에 급수 n= 10



```
  int i, cache = 0;
  glBegin(GL_LINES);
  glColor3f(1,0,1);
  for(x=-10*M_PI;x<=10*M_PI;x+=inc)
  {
    yp = 0;

    for(i = 1; i < 10; i++)
      yp += ((1.0 - cos(i * M_PI)) / (i * M_PI)) * sin(i * x);

    y = yp + 0.5;

    if(cache)
    {
      glVertex2f(cx, cy);
      glVertex2f(x, y);
    }

    cache = 1;
    cx = x;
    cy = y;
  }
  glEnd();
}

void originAxis(void)
{
  glBegin(GL_LINES);
  glColor3f(0,0,1);
  glVertex3f(-100,0,0);
  glVertex3f(100, 0, 0);
```

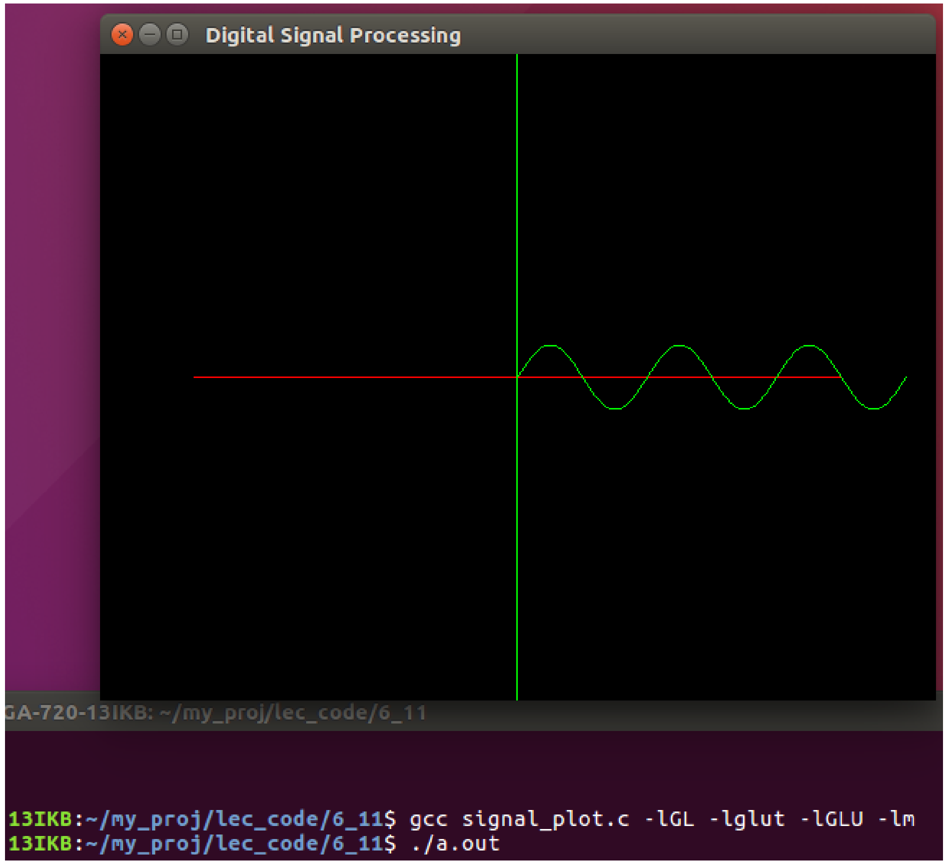
fourier 급수 n = 10000



```
  glColor3f(1,0,0);
  glVertex3f(0,-100,0);
  glVertex3f(0, 100, 0);
  glColor3f(0,0,1);
  glVertex3f(0,0,0);
  glVertex3f(0, 0, 1);
  glEnd();

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH);
  glutInitWindowSize(800, 800);
  glutCreateWindow("Fourier Series(Rectangular Wave)");

  glOrtho(-10 * M_PI, 10 * M_PI, -0.5, 1.3, -1.0, 1.0);
  glEnable(GL_DEPTH_TEST);

  glutDisplayFunc(display);
  glutMainLoop();

  return EXIT_SUCCESS;
}
```

5. signal_plot

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#include <GL/glut.h>

#define SLICE   360

void draw_omega_sin(void);

float common_angles[5] = {15.0, 30.0, 45.0, 60.0, 75.0};
float freq_table[5] = {1000.0, 2400.0, 5000.0, 24000.0, 77000.0};

float theta = 0.0;

void display(void)
{
  glClearColor(0.0, 0.0, 0.0, 1.0); // 설정 초기화 r,g,b, 투명도 1.0 ==> 불투명
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();

  //gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  glColor3f(1, 0, 0);    // 레드 x축 설정

  glBegin(GL_LINE_LOOP);
  glVertex3f(100.0, 0.0, 0.0);
  glVertex3f(-100.0, 0.0, 0.0);
```


Digital Signal Processing
GA-720-13IKB: ~/my_proj/lec_code/6_11

```
13IKB:~/my_proj/lec_code/6_11$ gcc signal_plot.c -lGL -lglut -lGLU -lm
13IKB:~/my_proj/lec_code/6_11$ ./a.out
```

```
  glEnd();

  glColor3f(0.0, 1.0, 0.0); // 그린 y축설정
  glBegin(GL_LINE_LOOP);
  glVertex3f(0.0, 100.0, 0.0);
  glVertex3f(0.0, -100.0, 0.0);
  glEnd();

  draw_omega_sin();
  glutSwapBuffers();
}

#if 0
void reshape(int w, int h)
{
  glViewport(0, 0, (GLsizei)w, (GLsizei)h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(60, (GLfloat)w / (GLfloat)h, 0.1, 100.0);
  glMatrixMode(GL_MODELVIEW);
}
#endif

void reshape(int w, int h)
{
    GLfloat n_range = 100.0f;

    if(h == 0)
        h = 1;

    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w <= h)
        glOrtho(-n_range, n_range, -n_range * h / w, n_range * h / w,
-n_range, n_range);
    else
        glOrtho(-n_range * w / h, n_range * w / h, -n_range, n_range,
-n_range, n_range);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
  switch(key)
  {
    case 27:
      exit(0);
      break;
  }
}

void set_rand_amplitude(float *amp)
{
  *amp = rand() % 3 + 3;
}

void set_angle_with_common_angles(float *angle)
{
  *angle = common_angles[rand() % 5];
}

void angle2radian(float *angle, float *radian)
{
  *radian = *angle * M_PI / 180.0;
}

void radian2angle(float *angle, float *radian)
{
  *angle = *radian * 180.0 / M_PI;
}

void set_rand_frequency(float *freq)
{
  *freq = freq_table[rand() % 5];
}

void calc_period(float *freq, float *period)
{
  *period = 1 / (*freq);
}

void calc_angular_velocity(float *freq, float *ang_vel)
{
  *ang_vel = 2 * M_PI * (*freq);
}

float get_step(float slice, float period)
{
  return period / slice;
}
```

```
void cos_sim(float amplitude, float ang_vel, float period)
{
  int cnt = 0;
  float step, t = 0.0;

  t = step = get_step(SLICE, period);

  while(cnt++ < 36)
  {
    printf("%.1fcos(%f * %.8f) = %f\n", amplitude, ang_vel,
      t, amplitude * cos(ang_vel * t));
    t += step;
  }
}

void sin_sim(float amplitude, float ang_vel, float period)
{
  int cnt = 0;
  float step, t = 0.0;

  t = step = get_step(SLICE, period);

  while(cnt++ < 36)
  {
    printf("%.1fsin(%f * %.8f) = %f\n", amplitude, ang_vel,
      t, amplitude * sin(ang_vel * t));
    t += step;
  }
}
```

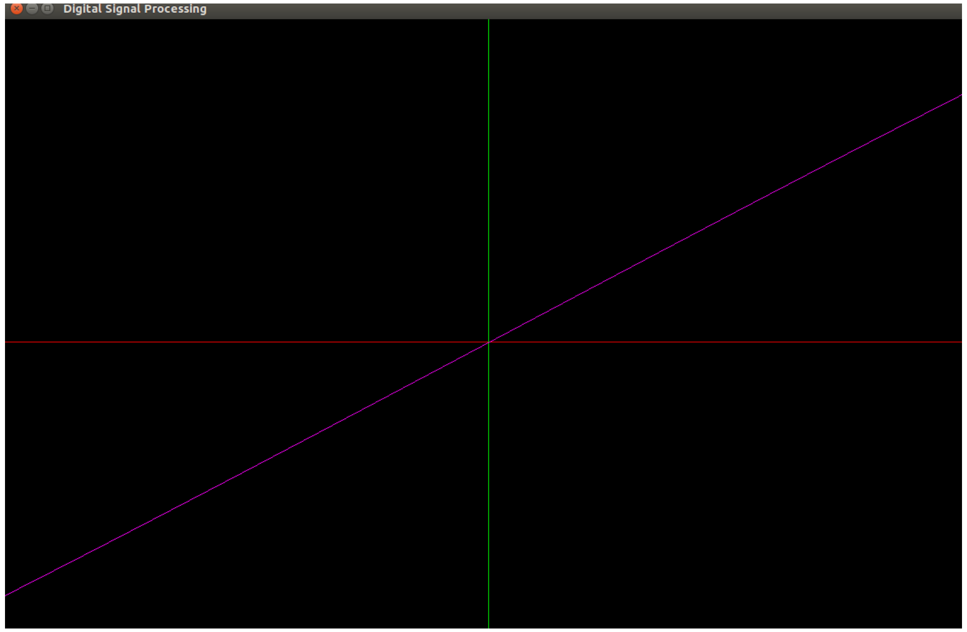
```
void draw_omega_sin(void)
{
   float amp, angle, period, freq, rad, omega, t, step = 0.0;
   float radius = 3.0;
   float x = 0, x2 = 0, y2, cx, cy;
   float tmp;
   int cache = 0;

   srand(time(NULL));

#if 0
   set_rand_amplitude(&amp);
   set_angle_with_common_angles(&angle);
   angle2radian(&angle, &rad);
   set_rand_frequency(&freq);
   calc_period(&freq, &period);
   calc_angular_velocity(&freq, &omega);
#endif

#if 1
   amp = 10;
   angle = 45.0;
   freq = 100.0;

   angle2radian(&angle, &rad); // radian 으로 바꿔줌
   calc_period(&freq, &period);
   calc_angular_velocity(&freq, &omega);  // 각속도 (신호처리에서는 각주파
수라고 부
름)
#endif
```

```
#if 0
  printf("amplitude = %f\n", amp);
  printf("angle = %f degree\n", angle);
  printf("radian = %f\n", rad);
  printf("frequency = %f\n", freq);
  printf("period = %f\n", period);
  printf("angular_velocity = %f\n", omega);
#endif

  t = step = get_step(SLICE, period); // 샘플링 주기

  //printf("t = %f\n", t);
#if 1
  if(t > period)
    t = 0.0;
#endif

  glBegin(GL_LINES);
  for(; ; t += step)
  {
    if(t > 3 * period)
    {
      break;
      t = 0.0;
    }

    //float rad_angle = angle * (M_PI / 180.0);
    //x2 += x;      // time += step;
    //x2 += 0.1;
    y2 = amp * sin(omega * t);
    //y2 = radius * sin((double)rad_angle);
```

```
      if(cache)
      {
         glVertex2f(cx * 4000, cy);    //단지 화면에 나오는 스케일 키운것, 너무 >
작아서
         glVertex2f(t * 4000, y2);
      }

      cache = 1;
      cx = t;
      cy = y2;
      //printf("t = %f, y2 = %f\n", t * 4000, y2);
   }
   glEnd();
}

int main(int argc, char **argv)
{
   float amplitude, angle, period, frequency, radian, angular_velocity;
   float step = 0.0;

   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE);
   glutInitWindowSize(1200, 800);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("Digital Signal Processing");

#if 0
   srand(time(NULL));

   set_rand_amplitude(&amplitude);
```

```
  set_angle_with_common_angles(&angle);
  angle2radian(&angle, &radian);
  set_rand_frequency(&frequency);
  calc_period(&frequency, &period);
  calc_angular_velocity(&frequency, &angular_velocity);

  printf("amplitude = %f\n", amplitude);
  printf("angle = %f degree\n", angle);
  printf("radian = %f\n", radian);
  printf("frequency = %f\n", frequency);
  printf("period = %f\n", period);
  printf("angular_velocity = %f\n", angular_velocity);

  cos_sim(amplitude, angular_velocity, period);
  sin_sim(amplitude, angular_velocity, period);
#endif

  glutDisplayFunc(display); // 동작을 빨리해줌
  //glutIdleFunc(display);
  glutReshapeFunc(reshape);
  //glutKeyboardFunc(keyboard);
  glutMainLoop();

  return 0;
}
```

6. fourier_series_line.c 선형성

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define _USE_MATH_DEFINES
#include <math.h>

#include <GL/glut.h>

#define SLICE   360

void rect_pulse_signal(void)
{
   float t, T = 0.01;
   float amp = 100;
   float step = 0.0;
   float omega = 2 * M_PI * 100;   // frequency = 5 Hz
   float x = 0, x2 = 0, y, cx, cy;
   int cache = 0;
   int i;

   //t = step = T / 100;
   step = T / 100;
   t = -1 * T;

   //printf("step = %f\n", step);

   if(t > T)
      t = 0.0;
```
	결과
	

```
  glColor3f(1.0, 0.0, 1.0);
  glBegin(GL_LINES);
  for(; ; t += step)
  {
    y = 0;

    if(t > 1 * T)
    {
      break;
      t = 0.0;
    }

    //y = amp * (sin(omega * t) / (omega * t));
    for(i = 1; i < 101; i++)
      y += 100 * ((1 - cos(i * M_PI)) / (i * M_PI) * sin(i * t));

    if(cache)
    {
      glVertex2f(cx * 6000, cy * 1);
      glVertex2f(t * 6000, y * 1);
    }
    cache = 1;
    cx = t;
    cy = y;
    //printf("t = %f, y = %f\n", t * 6000, y * 1);
  }
  glEnd();
}

void display(void)
```

```
{
  glClearColor(0.0, 0.0, 0.0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();

  //gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  glColor3f(1, 0, 0);

  glBegin(GL_LINE_LOOP);
  glVertex3f(100.0, 0.0, 0.0);
    glVertex3f(-100.0, 0.0, 0.0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);

    glBegin(GL_LINE_LOOP);
    glVertex3f(0.0, 100.0, 0.0);
    glVertex3f(0.0, -100.0, 0.0);
    glEnd();

  rect_pulse_signal();
  glutSwapBuffers();
}

void reshape(int w, int h)
{
    GLfloat n_range = 20.0f;

    if(h == 0)
        h = 1;
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w <= h)
        glOrtho(-n_range, n_range, -n_range * h / w, n_range * h / w,
-n_range, n_range);
    else
        glOrtho(-n_range * w / h, n_range * w / h, -n_range, n_range,
-n_range, n_range);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
  switch(key)
  {
    case 27:
      exit(0);
      break;
  }
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE);
  glutInitWindowSize(1200, 800);
  glutInitWindowPosition(0, 0);
  glutCreateWindow("Digital Signal Processing");

  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();

  return 0;
}
                                              130,1      바닥
```