2025-02-07-vglare

진행사항

- numerical한 방법이 아닌 적분 가능한 glare model 꼴을 찾아서 analytic한 1D fourier series 구현
- test를 시각화하기 위한 rex 구현

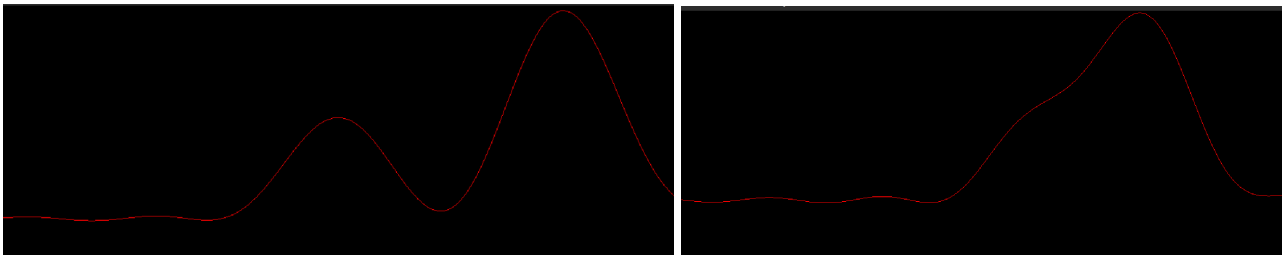
1D Fourier Series Expansion

어떤 함수 f(x)에 해서 fourier series로 나타내기 위해 각 cos, sin 항에 대한 계수 a_n, b_n이 필요하다. 근사하기 위한 구간[-L, L] 사이를 주기 $T(=\lambda) = 2L$ 이라고 했을 때, coefficient 식은 다음과 같다.

$$\omega = \frac{2\pi}{\lambda}, a_n = \frac{\tau}{\lambda}\int_{-L}^{L} f(x)\cos(n\omega x)dx$$

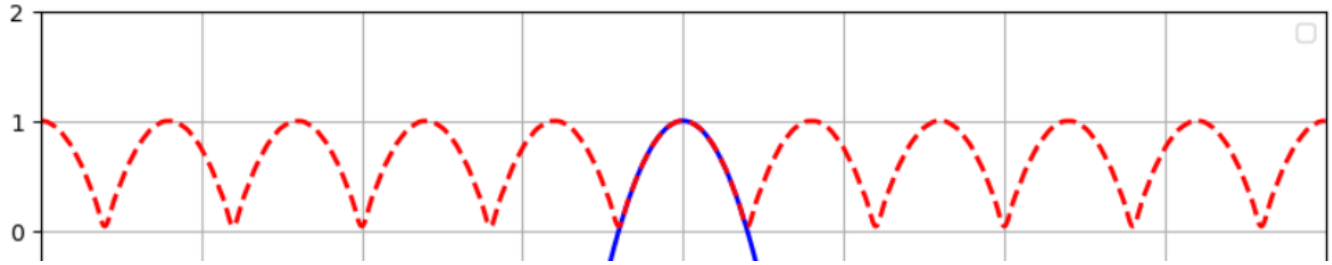
b_n도 동일한 방법으로 구할 수 있지만 해당 방법에는 문제점이 존재한다. $f(x)cos(n\omega x)$가 수학적으로 적분이 불가능하다면 해당 [-L, L] 구간을 N개 sampling 해서 $\sum$ 하는 방법으로 진행해야 한다. 기존 gaussian 역시 sampling 하여 fourier series로 만들었을 때, gaussian의 이동과 병합이 자유로운 것은 확인했지만 이전에 적분 가능한 함수에 대해 numerical하지 않은 방법을 테스트하기로 결정했다. 따라서 우선 glare model의 개형과 비슷하게 생기면서 analytic한 function을 찾아서 1D fourier series로 구현했다.

아래 사진은 2개 gaussian의 mean 값을 변경하면서 coefficents를 각각 구한 후 병합한 모습이다.



y = 1 - x^2

$y = 1 - x^2$ 함수를 [-1, 1] 구간에서 fourier series로 근사하면 위로 볼록한 함수를 얻을 수 있다. fourier series로 근사한 식 f'(x)를 [-10, 10]으로 확장하면 아래와 같은 그래프를 얻는다.



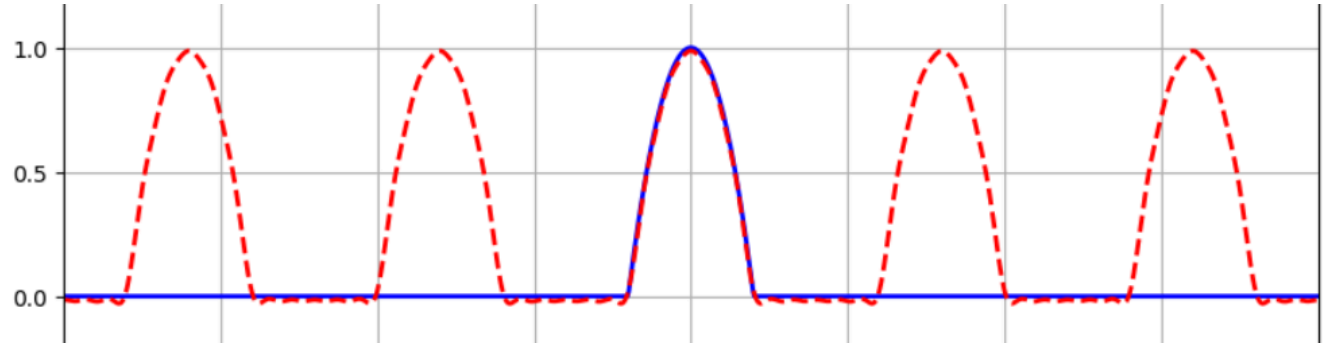
따라서 주기성이 없는 함수 f(x)를 [a, b] 구간에 대해서만 fourier series를 근사하면 f'(x)는 [a, b]에 대해서 반복되는 꼴을 얻는다는 것을 알 수 있다. 위 함수는 $1-x^2$을 [-1, 1]에 대해서 적분한 것으로 다음과 같은 식으로 세워진다.

$$a_0 = \int_{-1}^{1}(1-x)^2 dx = \frac{4}{3}, a_n = \int_{-1}^{1}(1-x)^2\cos(n\pi x)dx = \frac{4(-1)^n+1}{n^2\pi^2}$$

$f(x) = 1-x^2$에 대해 [-2, -1], [1, 2]에서 0인 직선 구간을 추가한다면 주기가 절반이고 구간마다 분리하여 적분해서 새로운 계수 식을 얻을 수 있다.

$$\begin{aligned} a_n &= \frac{1}{2}\int_{-2}^{2} f(x)cos(n\pi x/2)dx \\ &= \frac{1}{2}\int_{-1}^{1}(1-x^2)cos(\frac{n\pi x}{2})dx \\ &= -\frac{8\cos(\frac{n\pi}{2})}{n^2\pi^2} + \frac{16\sin(\frac{n\pi}{2})}{n^3\pi^3}(n \geq 1) \end{aligned} \qquad = -\frac{8\cos(\frac{n\pi}{2})}{n^2\pi^2} + \frac{16\sin(\frac{n\pi}{2})}{n^3\pi^3}$$

그래프는 다음과 같다.



general form

1. [-2, 2]에서 [-A, B]의 비대칭 경우(주기 T 조정)

예를 들어 [-2, 3] 구간에서 [-1, 1] 구간이 Ω shape를 가질 때.

$$\frac{2}{5}\int_{-1}^{1}(1-x^2)\cos\left(N\frac{2\pi}{5}x\right)dx$$

sin 항은 [-1, 1]에 대해서 0을 갖는다. 이 때 화면을 [A, B]를 잡아도 전체 주기로 보면 [-C, C] 처럼 대칭적인 모양이 나옴. 즉 mean = 0이고 주기가 T, (A^2-x^2) 어떤 구간을 잡아도 degree n인 계수는

$$
\begin{aligned}
a_0 &= \frac{4A^3}{3T} \\
a_n &= \frac{2}{T}\int_{-A}^{A}(A^2-x^2)\cos\left(n\frac{2\pi}{T}x\right)dx \\
&= \frac{8}{T}\left(\frac{\sin\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^3} - \frac{A\cos\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^2}\right), (n \geq 1)
\end{aligned}
$$

fourier coefficient를 구하기 위한 적분식의 analytic solution을 얻을 수 있다.

2. mean 값이 0이 아닌 경우(x축 이동)

(A^2-x^2)에서 $(A^2-(x-t)^2)$ 일 때 적분식의 closed-form solution을 구한다. 적분 식에 x가 들어가지 않기 때문에 coefficient 식에 변화가 없어서 간단하게 구할 수 있다.

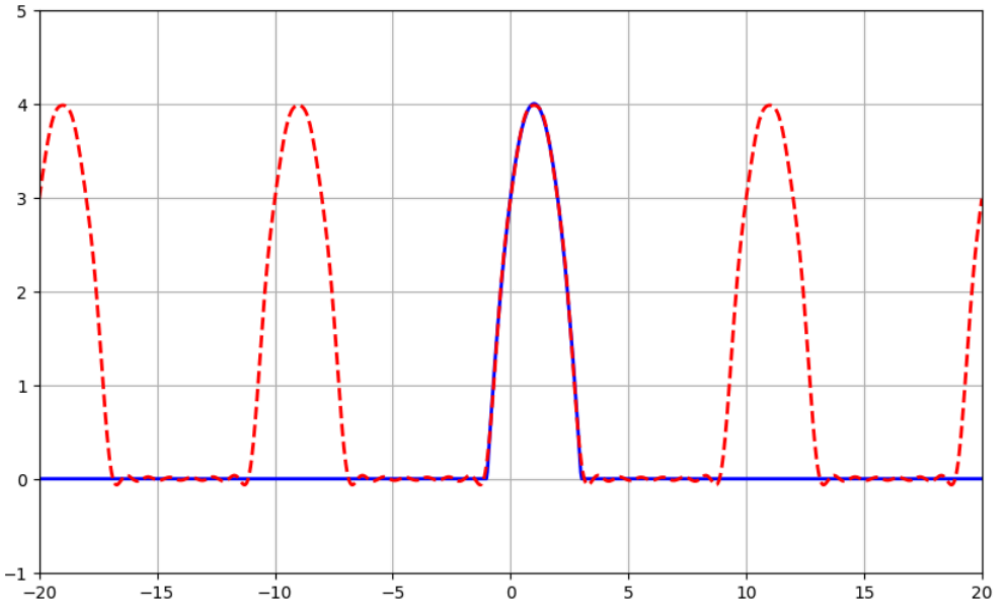
3. $1-x^2$ 꼴의 general한 glare model 표현

glare model의 주기는 T, shape(gaussian의 σ와 비슷한 역할) A, mean 위치를 t라고 놓았을 때 N-degree fourier series는 다음과 같다.

$$
\begin{aligned}
a_0 &= \frac{4A^3}{3T} \\
a_n &= \frac{2}{T}\int_{-A+t}^{A+t}(A^2-(x-t)^2)\cos\left(n\frac{2\pi}{T}x\right)dx \\
&= \frac{8}{T}\cos\left(\frac{2\pi n}{T}t\right)\left(\frac{\sin\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^3} - \frac{A\cos\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^2}\right), (n \geq 1) \\
b_0 &= 0 \\
b_n &= \frac{2}{T}\int_{-A+t}^{A+t}(A^2-(x-t)^2)\sin\left(n\frac{2\pi}{T}x\right)dx \\
&= \frac{8}{T}\sin\left(\frac{2\pi n}{T}t\right)\left(\frac{\sin\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^3} - \frac{A\cos\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^2}\right), (n \geq 1)
\end{aligned}
$$

$$
f(x) = \frac{4A^3}{3T} + \sum_{n=1}^{N} a_n\cos\left(\frac{2\pi n}{T}x\right) + b_n\sin\left(\frac{2\pi n}{T}x\right)
$$

예를 들어 A = 2, T = 10, t = 1인 식의 fourier series는 다음과 같이 나타난다. 파란색 실선이 원본 식, 붉은 점선이 fourier series로 근사한 식이다.



rex implementation

$$
\begin{aligned}
a_n &= \frac{1}{2}\int_{-2}^{2} f(x)cos(n\pi x/2)dx \\
&= \frac{1}{2}\int_{-1}^{1} (1-x^2)cos(\frac{n\pi x}{2})dx \\
&= -\frac{8\cos(\frac{n\pi}{2})}{n^2\pi^2} + \frac{16\sin(\frac{n\pi}{2})}{n^3\pi^3}(n \geq 1) \qquad = -\frac{8\cos(\frac{n\pi}{2})}{n^2\pi^2} + \frac{16\sin(\frac{n\pi}{2})}{n^3\pi^3}
\end{aligned}
$$

위의 기본적인 형태의 적분 식을 rex에서 실행해봤다. 적분 구간은 (-2.0,2.0)이며, fourier series N개의 항의 각각의 coefficient들은 float vector A, B에 저장했다.

```
struct REXAPI UserData : public GUIData
{
    gl::Framebuffer*    FBO=nullptr;
    gl::Effect*         effect=nullptr;
    gl::VertexArray* vao;


    vector<float> A;
    vector<float> B;
};
```

vao는 GLFrame을 통해 그릴 그래프의 각 지점을 vertex로 표현했을 때의 vertex array이다.

```
struct REXAPI UserFunc : public UserData
{
    vector<vertex> export_result();
```

```
};
```

initial_update()부분에서 모든 N개의 coefficient들을 계산하여 A,B배열에 저장했다. 해당 함수는 sin항은 모두 0이 되기 때문에 A만 계산했다.

```
bool RexPlugImpl::initial_update()
{
    if(!FBO) FBO = gxCreateFramebuffer("FBO"); if(!FBO) return false;
    safe_delete(effect)=gxCreateEffect( get_name(), STR_FS1D2_FX );
if(!effect) return false;

    std::function<float(int)> temp_func = [](int n) {
        return -8.0f * cos(n * PI<float> / 2.0f) / (n * n * PI<float> *
PI<float>) + 16.0f * sin(n * PI<float> / 2.0f) / (n * n * n * PI<float>
*PI<float> * PI<float>);
    };

    A = vector<float>(N + 1);
    B = vector<float>(N + 1);

    for(int n = 1; n <= N; n++)
    {
        A[n] += temp_func(n);
        B[n] = 0.0f;
    }

    return true;
}
```

```
    std::function<float(int)> temp_func = [](int n) {
        return -8.0f * cos(n * PI<float> / 2.0f) / (n * n * PI<float> *
PI<float>) + 16.0f * sin(n * PI<float> / 2.0f) / (n * n * n * PI<float>
*PI<float> * PI<float>);
    };
```

위의 함수는 아래의 적분 결과 식을 그대로 implement한 것이다. n항에 대해서 값이 다르게 나오기 때문에 parameter를 n으로 설정했다.

$$-\frac{8\cos(\frac{n\pi}{2})}{n^2\pi^2}+\frac{16\sin(\frac{n\pi}{2})}{n^3\pi^3}$$

```
    A = vector<float>(N + 1);
    B = vector<float>(N + 1);
```

```
for(int n = 1; n <= N; n++)
{
    A[n] += temp_func(n);
    B[n] = 0.0f;
}
```

전체 ghost의 n항 coefficient를 A[n], B[n]에 저장한다.

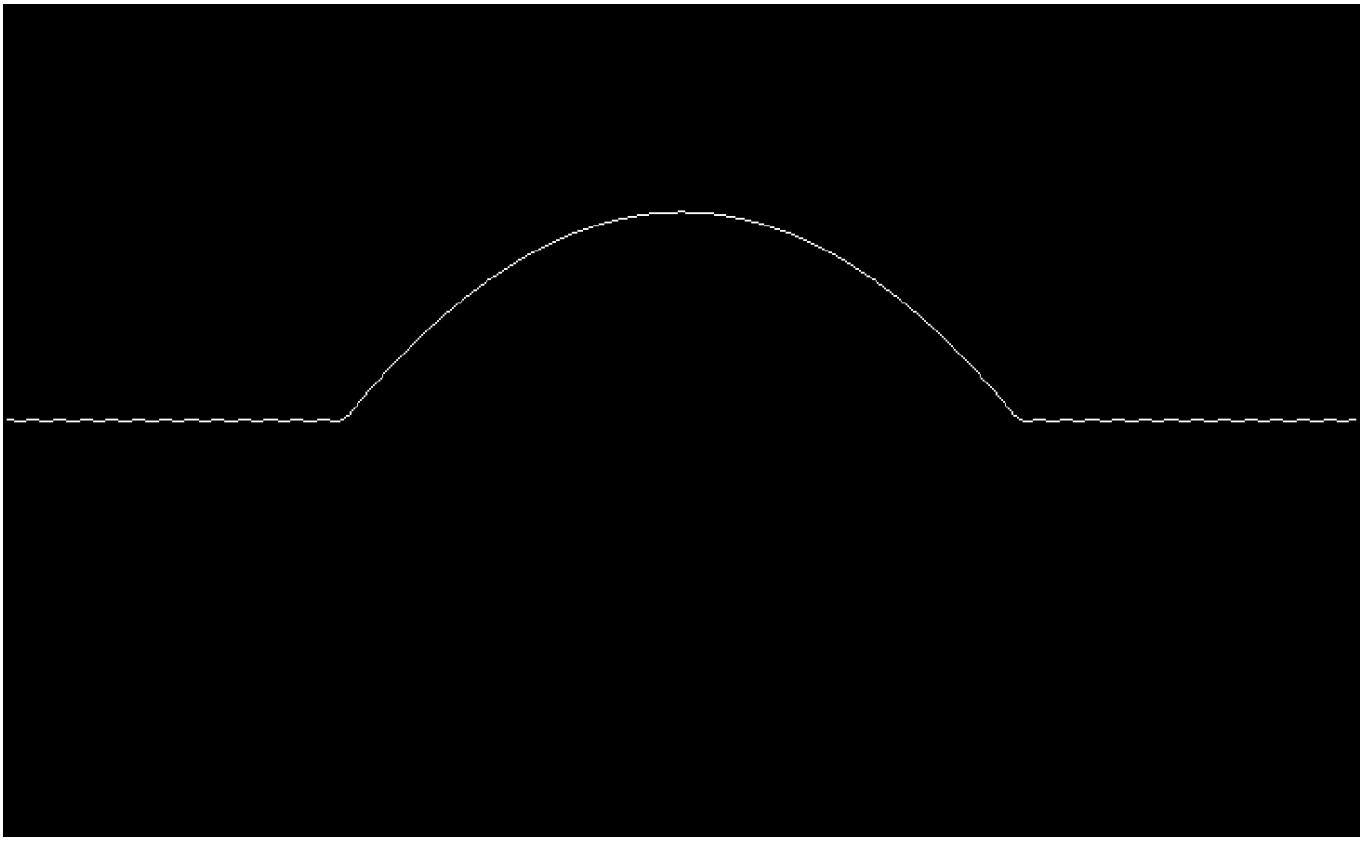
```
vector<vertex> vertices = export_result();
vao = gxCreateVertexArray("LINES", vertices.data(), vertices.size());
if(!vao) return;

FBO->set_state(false, false);
FBO->bind(DST);
FBO->clear();

effect->bind( "draw_fs" );
vao->draw_arrays(0, 0, GL_LINE_STRIP);
```

export_result()함수를 통해 A, B coefficients를 cos과 sin에 곱하여 나온 결과값을 vertex로 저장한다.

```
vector<vertex> vertices;
float omega = 2.0f * PI<float> / 4.0f;

for(float x = -2.0f; x <= 2.0f; x += 0.01f)
{
    float sum = 0.0f;

    for(int n = 0; n <= N; n++)
    {
        sum += A[n] * cos(n * omega * x) + B[n] * sin(n * omega * x);
    }

    vertex v = {vec3(x / 2.0f, sum, 0), vec3(1.0f, 0, 0), vec2(0)};
    vertices.push_back(v);
}

return vertices;
```

해당 함수의 주기를 4로 설정했기 때문에 omega값을 위와 같이 계산한다. 모든 계산하는 x에 대하여 0부터 N까지의 cos, sin항에 pre-computed된 coefficients를 곱하여 누적함으로써 y값을 얻

어낸다. 이를 vertex array에 추가하여 반환한다.



general form rex implementation

추가적으로, 함수를 x축 방향으로 t만큼 평행 이동시켰을 때, 봉우리의 peak를 A로 설정했을 때, 주기가 T일 때 general 하게 대응할 수 있도록, 변수 t(평행 이동), T(주기), A(peak)를 적용한 general한 함수를 구성했다.
공식은 위에 정리한 수학 식을 사용하며, 평행 이동이 들어가기 때문에 cos뿐만 아니라 sin항의 coefficient까지 모두 계산해야 되며, 다음의 적분 식을 적용한다.

$$
\begin{aligned}
a_0 &= \frac{4A^3}{3T} \\
a_n &= \frac{2}{T}\int_{-A+t}^{A+t}(A^2-(x-t)^2)\cos\left(n\frac{2\pi}{T}x\right)dx \\
&= \frac{8}{T}\cos\left(\frac{2\pi n}{T}t\right)\left(\frac{\sin\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^3}-\frac{A\cos\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^2}\right), (n\geq 1)
\end{aligned}
$$

$$
\begin{aligned}
b_0 &= 0 \\
b_n &= \frac{2}{T}\int_{-A+t}^{A+t}(A^2-(x-t)^2)\sin\left(n\frac{2\pi}{T}x\right)dx \\
&= \frac{8}{T}\sin\left(\frac{2\pi n}{T}t\right)\left(\frac{\sin\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^3}-\frac{A\cos\left(\frac{2\pi nA}{T}\right)}{\left(\frac{2\pi n}{T}\right)^2}\right), (n\geq 1)
\end{aligned}
$$

이를 그대로 rex에 적용했다.

```
    std::function<float(float, float, float, int)> temp_func_a = [](float A,
float T, float t, int n) {
        float c = 2 * PI<float> *n / T;
        return 8.0f / T * cos(c * t) * (sin(c * A) / (c * c * c) - A * cos(c *
A) / (c * c));
    };

    std::function<float(float, float, float, int)> temp_func_b = [](float A,
float T, float t, int n) {
        float c = 2 * PI<float> *n / T;
        return 8.0f / T * sin(c * t) * (sin(c * A) / (c * c * c) - A * cos(c *
A) / (c * c));
    };
```

A,T,t,n값에 따라 A, B coefficient를 구하는 함수를 생성했다.
k개의 ghost에 대해 각 함수의 A(peak), T(period), t(offset)을 vector로 저장하여 p_A, p_T, p_t로 전달했다.

```
    p_A = vector<float>(K);
    p_T = vector<float>(K);
    p_t = vector<float>(K);
    p_A[0] = 0.5f;
    p_A[1] = 1.0f;
    p_T[0] = 4.0f;
    p_T[1] = 4.0f;
    p_t[0] = 0.0f;
    p_t[1] = 0.0f;
```

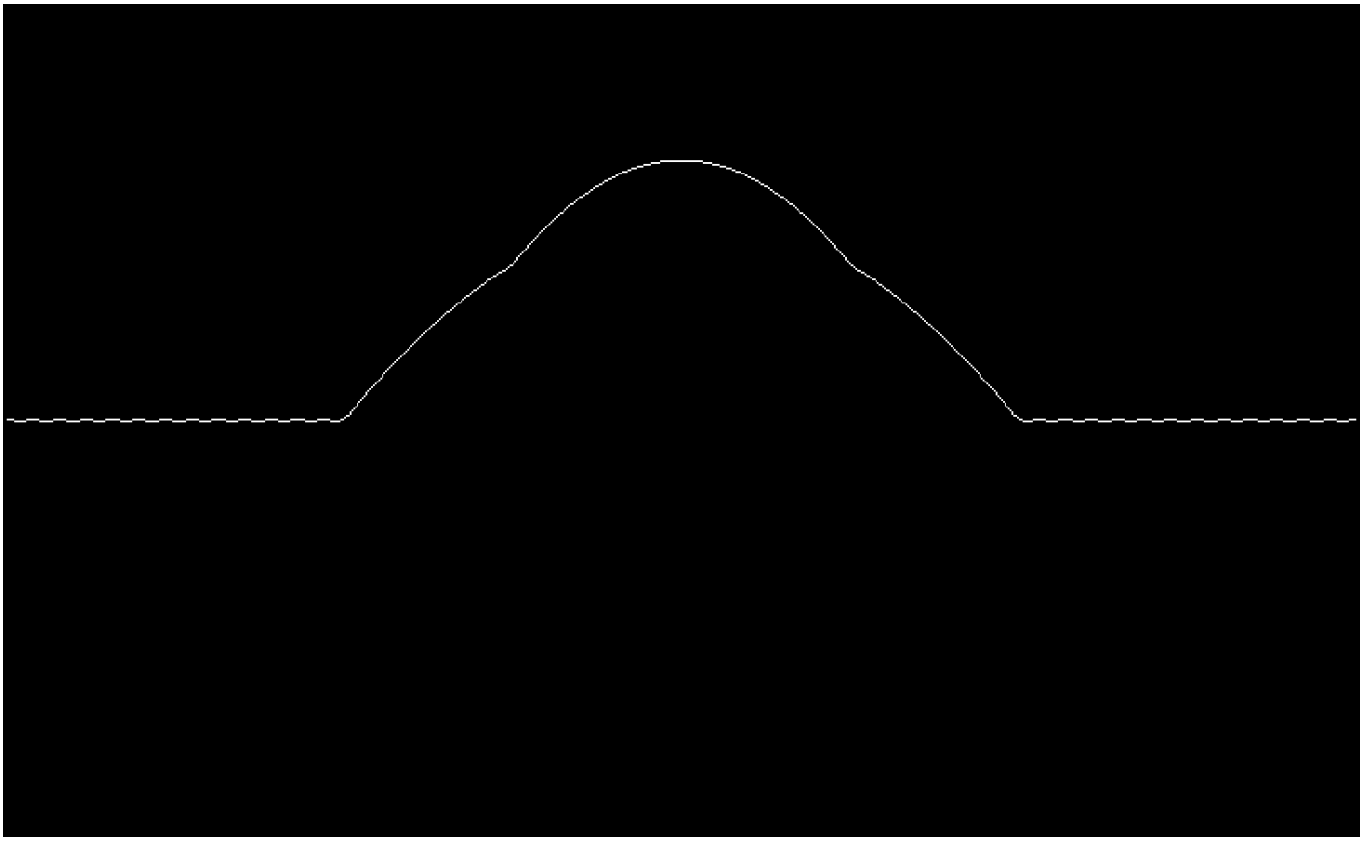
모든 K ghost의 coefficient를 다음과 같이 aggregation했다.

```
    A = vector<float>(N + 1);
    B = vector<float>(N + 1);

    for(int k = 0; k < K; k++)
    {
        A[0] += 4.0f * p_A[k] * p_A[k] * p_A[k] / (3.0f * p_T[k]);
    }

    for(int n = 1; n <= N; n++)
```

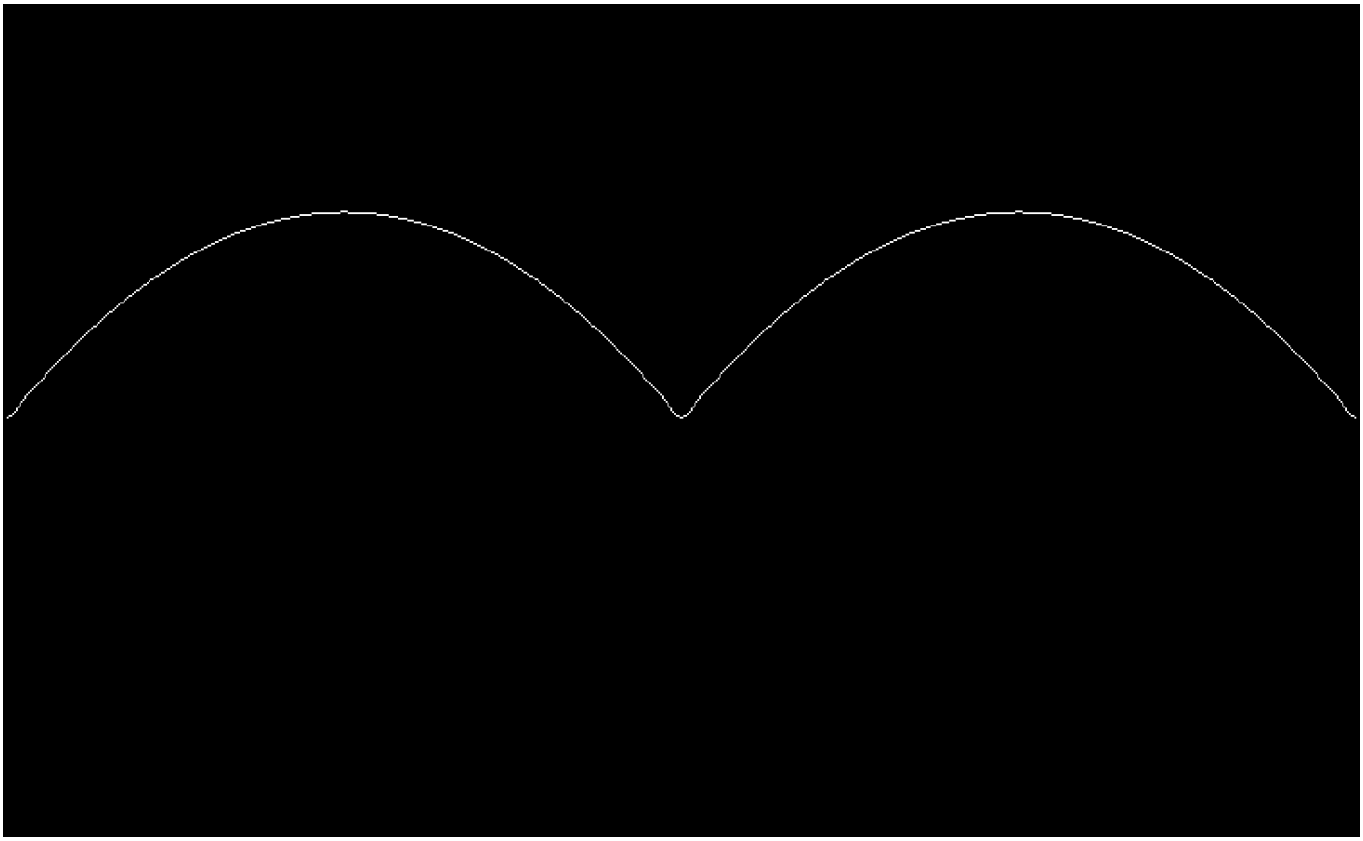
```
    {
        for(int k = 0; k < K; k++)
        {
            A[n] += temp_func_a(p_A[k], p_T[k], p_t[k], n);
            B[n] += temp_func_b(p_A[k], p_T[k], p_t[k], n);
        }

    }
```

N+1개의 항을 순차적으로 각 항마다 K개의 ghost의 coefficient를 sum했다.

함수의 중심이 0.0, 0.0이며 peak가 0.5, 1.0인 두 함수를 aggregation하도록 2개의 fourier series coefficient를 합쳐서 전체 coefficient 배열에 저장했다.
그 후, 위와 같은 방식으로 함수를 그래프로 그리면 다음과 같다.



함수의 중심이 -1.0, 1.0이며 peak가 1.0 1.0인 두 개의 함수를 합쳤다.



x, y 축을 모두 (-2.0, 2.0) 범위를 정규화 하여 그래프로 그렸다.