# Selected files

**3 printable files**

pnn_io.c
pnn.c
pnn.h

**pnn_io.c**

```c
#include "pnn.h"

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

void pnn_fread(pnn * nn, FILE * stream)
{
    fread(&(nn->sigma), sizeof(double), 1, stream);

    int rfv_count;
    fread(&rfv_count, sizeof(int), 1, stream);
    nn->rfv_count = rfv_count;

    nn->refs = (v2 *)malloc(sizeof(v2) * rfv_count);
    nn->f_vals = (double *)malloc(sizeof(double) * rfv_count);
    fread(nn->refs, sizeof(v2), rfv_count, stream);
    fread(nn->f_vals, sizeof(double), rfv_count, stream);
}

void pnn_fwrite(pnn * nn, FILE * stream)
{
    fwrite(&(nn->sigma), sizeof(double), 1, stream);
    fwrite(&(nn->rfv_count), sizeof(int), 1, stream);

    fwrite(nn->refs, sizeof(v2), nn->rfv_count, stream);
    fwrite(nn->f_vals, sizeof(double), nn->rfv_count, stream);
}

void pnn_fprint(pnn * nn, FILE * stream, int head_count)
{
    fprintf(stream, "sigma = %lf; rfv_count = %d\n", nn->sigma, nn->rfv_count);

    for (int i = 0; (i < nn->rfv_count) && (i < head_count); i++)
        fprintf(stream, "ref #%d: x = %lf, y = %lf | f value #%d: %lf\n",
                i, nn->refs[i].x, nn->refs[i].y, i, nn->f_vals[i]);
}

```

**pnn.c**

```c
#include "pnn.h"

#include <stdlib.h>
```

```c
#include <math.h>

static double act(v2 * x, v2 * ref, double sigma);
static double eucl2(v2 * a, v2 * b);

void pnn_new(double sigma,
             double a, double b, int segment_count,
             pnn * nn,
             double(*f)(double, double))
{
    int rfv_count = segment_count * segment_count;

    nn->sigma = sigma;
    nn->rfv_count = rfv_count;
    nn->refs = (v2 *)malloc(sizeof(v2) * rfv_count);
    nn->f_vals = (double *)malloc(sizeof(double) * rfv_count);

    double delta = (b - a) / (double)segment_count;
    double x, y;
    int k;
    for (int i = 0; i < segment_count; i++)
    {
        x = a + (double)i * delta;
        for (int j = 0; j < segment_count; j++)
        {
            y = a + (double)j * delta;

            k = i * segment_count + j;

            nn->refs[k].x = x;
            nn->refs[k].y = y;
            nn->f_vals[k] = f(x, y);
        }
    }
}

double pnn_run(pnn * nn, v2 * x)
{
    double result = 0;

    for (int i = 0; i < nn->rfv_count; i++)
        result += act(x, &(nn->refs[i]), nn->sigma) * nn->f_vals[i];

    return result;
}

static double act(v2 * x, v2 * ref, double sigma)
{
    return exp((-1.0 * (eucl2(x, ref)) / (sigma * sigma)));
}

static double eucl2(v2 * a, v2 * b)
{
    double x = b->x - a->x;
    double y = b->y - a->y;
    return x * x + y * y;
}

```

**pnn.h**

```c
#pragma once

#include <stdio.h>

typedef struct
{
    double x;
    double y;
} v2;

typedef struct
{
    double sigma;
    int rfv_count;
    v2 * refs;
    double * f_vals;
} pnn;

void pnn_new(double sigma,
             double a, double b, int segment_count,
             pnn * nn,
             double(*f)(double, double));

double pnn_run(pnn * nn, v2 * x);

void pnn_fread(pnn * nn, FILE * stream);

void pnn_fwrite(pnn * nn, FILE * stream);

void pnn_fprint(pnn * nn, FILE * stream, int head_count);

```