```c
#include "useful_func.h"

float map(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

void map_vec_to_dac(float* dom, uint16_t* range, size_t N){
    for(int i=0; i<N; i++){
        range[i] = (uint16_t) map(dom[i], -1, 1, 0, 4095);
    }
}

float* zeros(float x[], size_t n){
  for(int i=0; i<n; i++){
    x[i] = 0;
  }
  return x;
}

void zeros_int(uint16_t x[], size_t n){
  for(int i=0; i<n; i++){
    x[i] = 0;
  }
}

void zeros2d(size_t row, size_t col, float x[][col]){
  for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){
      x[i][j] = 0;
    }
  }
}

void generate_harmonics(size_t rows, size_t col, float harmonics[][col], float f, uint8_t K, float* Ck, float* Ok){

  float w = 2*PI*f;
  float Fs = 12*f*K;
  float T = 1/Fs;
  size_t interval = 12*K;

  for(int i=0; i<K; i++){
    for(int n=0; n<interval; n++){
      harmonics[i][n] = Ck[i]*cosf((i+1)*w*n*T + Ok[i]);
    }
  }

}

void add_vecs(float* vec1, float* vec2, float* sum, size_t N){
  for(int i=0; i<N; i++){
    sum[i] = vec1[i] + vec2[i];
  }
}

void add_const_vec(float k, float* vec, size_t N){
  for(int i=0; i<N; i++){
    vec[i] += k;
  }
}
```