/\*

\* math\_formula.c

\*

\* Created on: Jul 14, 2023

\* Author: udin

\*/

#include "math\_formula.h"

pr m[6] = { //motor 1

{.c0 = -5.0691673081360022e-012, .c1 = 1.5957142857153541e+000, .c2 = -3.8571428571550128e-003

,.c3 = 9.8571428571836824e-006, .c4 = -8.5714285714700861e-009}, //rf

{.c0 = -9.7764019102442035e-012, .c1 = 1.7242857142877093e+000, .c2 = -5.0011904762130811e-003

,.c3 = 1.3892857142932903e-005, .c4 = -1.3095238095315033e-008}, //lf

{.c0 = -7.1500583231909332e-012, .c1 = 1.5760317460332156e+000, .c2 = -4.3658730158896951e-003

,.c3 = 1.2182539682595640e-005, .c4 = -1.1269841269898140e-008}, //lb

{.c0 = -7.3461237093397358e-012, .c1 = 1.7126190476206040e+000, .c2 = -4.7928571428748510e-003

,.c3 = 1.2809523809583281e-005, .c4 = -1.1428571428631899e-008}, //rb

{.c0 = -9.1706642280087181e-012, .c1 = 1.8608730158749087e+000, .c2 = -5.4281746031960964e-003

,.c3 = 1.4519841269913400e-005, .c4 = -1.3253968254041571e-008}, //ll

{.c0 = -7.3324679661368464e-012, .c1 = 1.6055555555570582e+000, .c2 = -4.0111111111281548e-003

,.c3 = 1.0444444444501601e-005, .c4 = -8.8888888889469583e-009} //rr

};

SStep a;

//result dalam hanya nilai positif (fabs)

float polyregress(pr \*pr, const float input){

float out\_ =

pr->c4 \* fabs(input) \* fabs(input) \* fabs(input) \* fabs(input) +

pr->c3 \* fabs(input) \* fabs(input) \* fabs(input) +

pr->c2 \* fabs(input) \* fabs(input) +

pr->c1 \* fabs(input) +

pr->c0 \* 1.0f;

return close\_to\_zero(out\_);

}

// invoke float hasil = smooth\_step(&a, t, sstep\_3th);

// result 0 < x < 1 untuk smoothing

float smooth\_step(SStep \*s, float in, order ord){

float range = (fabs(in) - s->min) / (s->max - s->min);

s->in = clamp(range, 0, 1);

float x\_ = ord(s->in) \* 1.0f;

return close\_to\_zero(x\_);

}

float sstep\_1th(float x){

return x \* x \* (3.0f - 2.0f \* x) / 1.0f;

}

float sstep\_2th(float x){

return x \* x \* x \* (x \* (6.0f \* x - 15.0f) + 10.0f) / 1.0f;

}

float sstep\_3th(float x){

return (-20.0f \* pow(x, 7) + 70.0f \* pow(x, 6) - 84.0f \* pow(x, 5) + 35.0f \* pow(x, 4));

}

float lin\_interp(float x, float in\_min, float in\_max, float out\_min, float out\_max){

float clamp\_x = clamp(x, in\_min, in\_max);

float out\_interp = (clamp\_x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + (out\_min);

float clamp\_out\_ = clamp(out\_interp, out\_min, out\_max);

return clamp\_out\_;

}

float close\_to\_zero(float value){

if(fabs(value - 0) <= ATOL + (RTOL \* fabs(value))) return 0.0f;

else return value \* 1.0f;

}

float clamp(float val, float min, float max){

if(val > max) return max \* 1.0f;

else if(val < min) return min \* 1.0f;

else return val \* 1.0f;

}