CFD

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <fstream>
using namespace std;

int inlet_velocity = 5;

void boundary_condition(vector<float>& velocity_vec, float length, float dx) {
    velocity_vec[0] = inlet_velocity;
    int size = velocity_vec.size() -1;
    int x2 = length;
    velocity_vec[size] = velocity_vec[size - 1] + (float)1/2 * (length + 3) * dx;
}

float calcu_err(vector<float> vec1, vector<float> vec2) {
    if (vec1.size() != vec2.size()) exit(-1);
    float sum = 0;
    for (int i = 0; i < vec1.size(); i++) {
        sum +=abs(vec2[i] - vec1[i]);
    }
    return sum;
}

void writefile(vector<float> velocity, vector<float> error, float dx) {
    std::ofstream velocity_data("velocity_data.csv");
    std::ofstream error_data("error_data.csv");

    if (!velocity_data) {
        cout << "error when opening the file" << endl;
        exit(0);
    }
    for (int i = 0; i < velocity.size(); i++) {
        velocity_data  << i << ","<< dx * i << "," << velocity[i] << endl;
    }
    for (int i = 0; i < error.size(); i++) {
```

```
        error_data << i << "," << error[i] << endl;
    }
    velocity_data.close();
    error_data.close();
}

int main()
{
    float max_error = 0.00001;
    int xelem_num = 10;
    float length = 1;

    float dx = length / xelem_num;
    float error;

    vector<float> velocity_vec(xelem_num +1, 0);
    vector<float> loc_vec;
    vector<float> error_data;

    for (int i = 0; i < xelem_num + 1; i++) loc_vec.push_back(i * dx);

    do{
        boundary_condition(velocity_vec,length, dx);
        vector<float> new_velocity_vec(velocity_vec);
        for (int i = 1; i < xelem_num; i++) {
            new_velocity_vec[i] = (float)1/2 * loc_vec[i] * dx * dx
            + (float)1/2 * (velocity_vec[i-1] + velocity_vec[i+1])
            - (float)1/8 * dx *(loc_vec[i] * loc_vec[i] + 5) * (velocity_vec[i + 1] - veloci
        }
        error = calcu_err(velocity_vec, new_velocity_vec);
        error_data.push_back(error);
        velocity_vec = new_velocity_vec;
    }while (error >= max_error);

    for (int i = 0; i < xelem_num + 1; i++) {
        cout << i << " " << velocity_vec[i] << endl;
    }

    vector<float> prof_vec1(xelem_num + 1,0);
    vector<float> prof_vec2(xelem_num + 1,0);
    for (int i = 1; i < xelem_num; i++) {
        prof_vec1[i] = (velocity_vec[i + 1] - velocity_vec[i - 1]) / (2 * dx);
        prof_vec2[i] = (velocity_vec[i + 1] + velocity_vec[i - 1] - 2 * velocity_vec[i]) / (
        float x = loc_vec[i];
        cout << "calculated: " << (x * x + 5) * prof_vec1[i] - 2 * prof_vec2[i] << ";  predi
    }
```

```
    writefile(velocity_vec, error_data, dx);
    return 0;
}
Copy
```