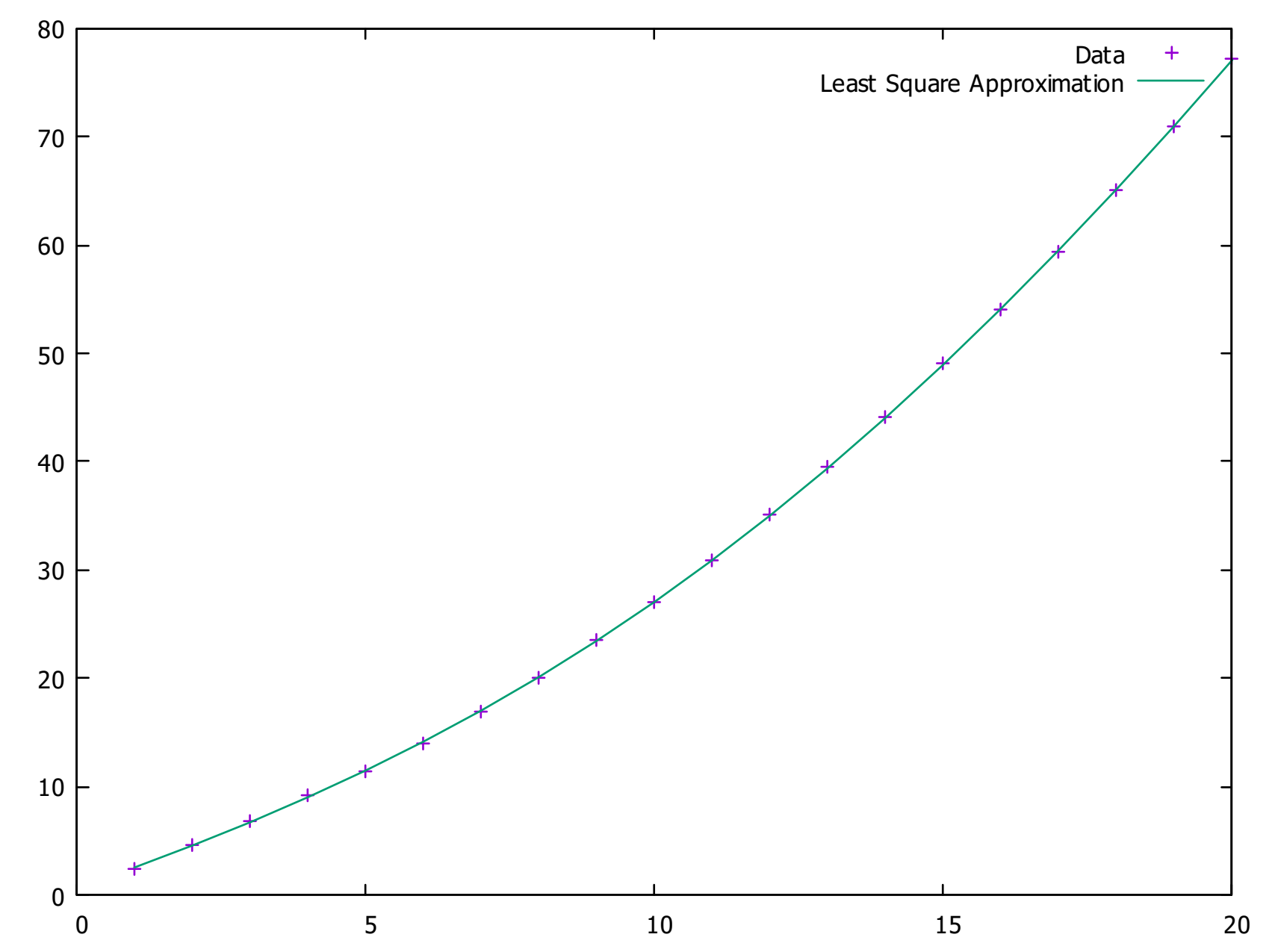
```cpp
#include <bits/stdc++.h>
using namespace std;

#ifdef WIN32
#define GNUPLOT_NAME "c:\\gnuplot\\bin\\gnuplot -persist"
#else
#define GNUPLOT_NAME "gnuplot -persist"
#endif

class ColumnVector {
public:
    int n;
    vector<double> v;
    ColumnVector(int m) {
        v = vector<double>(m, 0);
        n = m;
    }
    void operator=(ColumnVector a) {
        for (int i = 0; i < n; i++) {
            v[i] = a.v[i];
        }
    }
};

void plotPointsAndCurve(vector<double> ts, vector<double> bs, vector<double> curve) {
#ifdef WIN32
    FILE* pipe = _popen(GNUPLOT_NAME, "w");
#else
    FILE* pipe = popen(GNUPLOT_NAME, "w");
#endif
    if (pipe != NULL) {
        fprintf(pipe, "%s\n", "plot '-' with points title 'Data', '-' with lines title 'Least Square Approximation'");
        // Plot points
        for (size_t i = 0; i < ts.size(); i++) {
            fprintf(pipe, "%f %f\n", ts[i], bs[i]);
        }
        fprintf(pipe, "e\n");
        // Plot regression polynomial
        for (size_t i = 0; i < curve.size(); i++) {
            fprintf(pipe, "%f %f\n", ts[i], curve[i]);
        }
        fprintf(pipe, "e\n");
        fflush(pipe);
#ifdef WIN32
        _pclose(pipe);
#else
        pclose(pipe);
#endif
    } else {
        cout << "Could not open pipe" << endl;
    }
}

ColumnVector LeastSquareApproximation(int n, int deg, vector<double> ts, vector<double> bs) {
    // Construct the matrix A
    vector<vector<double>> A(n, vector<double>(deg + 1, 0));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j <= deg; ++j) {
            A[i][j] = pow(ts[i], j);
        }
    }
    // Construct the matrix B
    vector<vector<double>> B(n, vector<double>(1, 0));
    for (int i = 0; i < n; ++i) {
        B[i][0] = bs[i];
    }
    // Transpose of matrix A
    vector<vector<double>> A_trans(deg + 1, vector<double>(n, 0));
    for (int i = 0; i <= deg; ++i) {
        for (int j = 0; j < n; ++j) {
            A_trans[i][j] = A[j][i];
        }
    }
    // Calculate (A^T * A)
    vector<vector<double>> ATA(deg + 1, vector<double>(deg + 1, 0));
    for (int i = 0; i <= deg; ++i) {
        for (int j = 0; j <= deg; ++j) {
            for (int k = 0; k < n; ++k) {
                ATA[i][j] += A_trans[i][k] * A[k][j];
            }
        }
    }

    // Calculate (A^T * B)
    vector<vector<double>> ATB(deg + 1, vector<double>(1, 0));
    for (int i = 0; i <= deg; ++i) {
        for (int j = 0; j < n; ++j) {
            ATB[i][0] += A_trans[i][j] * B[j][0];
        }
    }
```

```
    // Perform LU decomposition on ATA
    vector<vector<double>> L(deg + 1, vector<double>(deg + 1, 0));
    vector<vector<double>> U(deg + 1, vector<double>(deg + 1, 0));
    for (int i = 0; i <= deg; i++) {
        for (int j = i; j <= deg; j++) {
            if (i == 0) {
                U[i][j] = ATA[i][j];
            } else {
                double sum = 0;
                for (int k = 0; k < i; k++) {
                    sum += L[i][k] * U[k][j];
                }
                U[i][j] = ATA[i][j] - sum;
            }
        }
        L[i][i] = 1;
        for (int j = i + 1; j <= deg; j++) {
            double sum = 0;
            for (int k = 0; k < i; k++) {
                sum += L[j][k] * U[k][i];
            }
            L[j][i] = (ATA[j][i] - sum) / U[i][i];
        }
    }

    // Solve for Y in LY = B
    vector<vector<double>> Y(deg + 1, vector<double>(1, 0));
    for (int i = 0; i <= deg; i++) {
        double sum = 0;
        for (int j = 0; j < i; j++) {
            sum += L[i][j] * Y[j][0];
        }
        Y[i][0] = (ATB[i][0] - sum) / L[i][i];
    }

    // Solve for coefficients in UX = Y
    vector<vector<double>> coefficients(deg + 1, vector<double>(1, 0));
    for (int i = deg; i >= 0; i--) {
        double sum = 0;
        for (int j = i + 1; j <= deg; j++) {
            sum += U[i][j] * coefficients[j][0];
        }
        coefficients[i][0] = (Y[i][0] - sum) / U[i][i];
    }

    ColumnVector result(deg + 1);
    for (int i = 0; i <= deg; ++i) {
        result.v[i] = coefficients[i][0];
    }
    return result;
}

int main() {
    // Prepare your own data set
    vector<double> ts = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};
    vector<double> bs = {2.5, 4.6, 6.9, 9.2, 11.5, 14.1, 16.9, 20.1, 23.5, 27.1, 30.9, 35.1, 39.5, 44.1, 49, 54.1, 59.4, 65.1, 71, 77.1};


    // Degree of the polynomial for regression
    int degree = 4;

    // Perform Least Square Approximation
    ColumnVector curveCoefficients = LeastSquareApproximation(ts.size(), degree, ts, bs);

    // Calculate the regression polynomial curve
    vector<double> curve(ts.size());
    for (size_t i = 0; i < ts.size(); i++) {
        double y = 0;
        for (int j = 0; j <= degree; j++) {
            y += curveCoefficients.v[j] * pow(ts[i], j);
        }
        curve[i] = y;
    }

    // Plot the graph of the point cloud and regression polynomial
    plotPointsAndCurve(ts, bs, curve);

    return 0;
}
```


Data
Least Square Approximation
80
70
60
50
40
30
20
10
0
0
5
10
15
20

1 Dataset

Here is the dataset used:

X	Y
1.0	2.5
2.0	4.6
3.0	6.9
4.0	9.2
5.0	11.5
6.0	14.1
7.0	16.9
8.0	20.1
9.0	23.5
10.0	27.1
11.0	30.9
12.0	35.1
13.0	39.5
14.0	44.1
15.0	49
16.0	54.1
17.0	59.4
18.0	65.1
19.0	71
20.0	77.1

a degree of 4 was used.