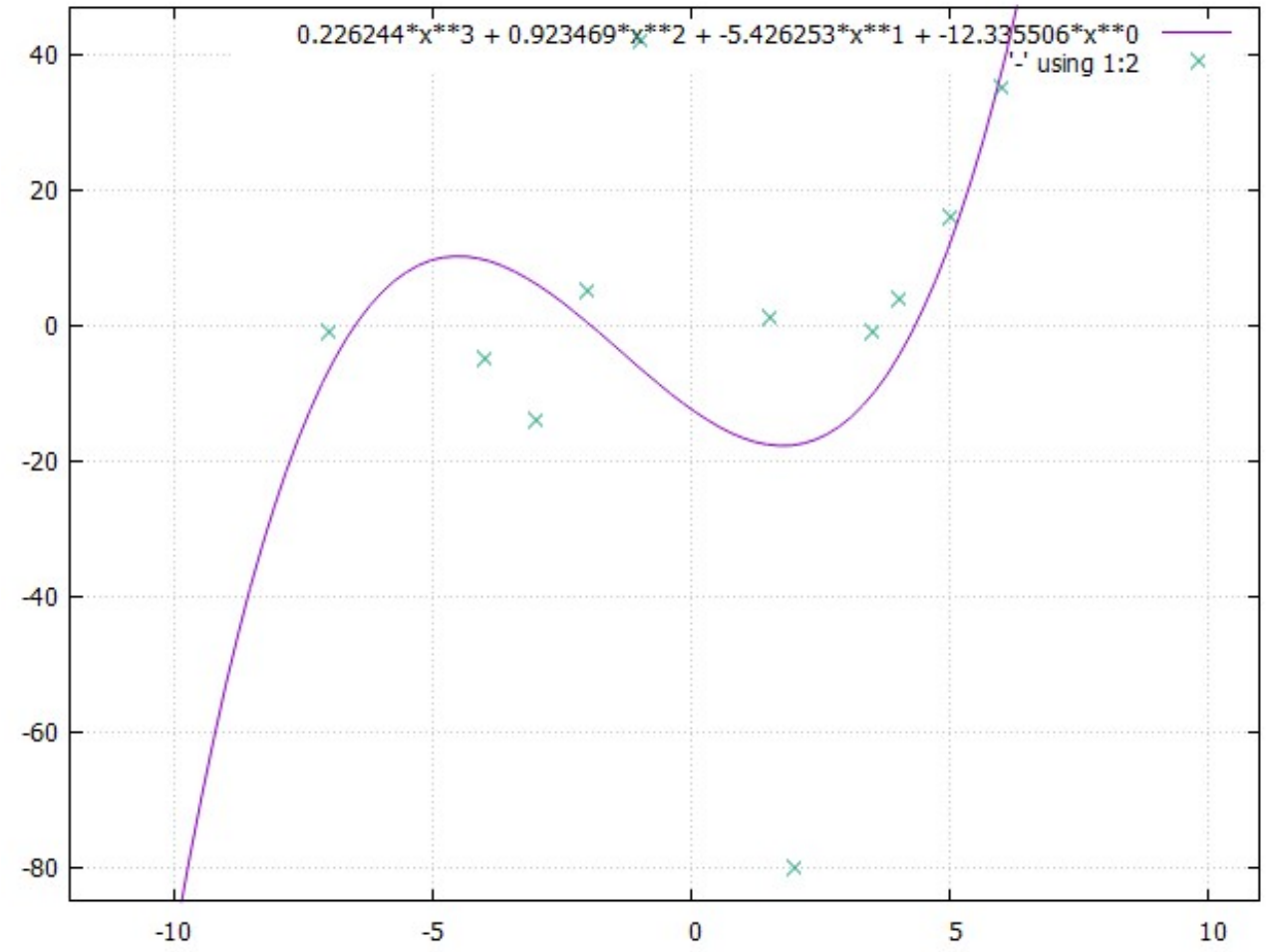
Analytical Geometry and Linear Algebra II

Dmitrii Kuzmin

April, 2023

DSAI-03
dm.kuzmin@innopolis.university

Plot:



Experimental data:

matrix_size = 11, polynomial_degree = 3

t	b
4.0	4.0
1.5	1.0
3.5	-1.0
-2.0	5.0
-7.0	-1.0
2.0	-80.0
5.0	16.0
6.0	35.0
-1.0	42.0
-3.0	-14.0
-4.0	-5.0

Main function (with gnuplot):

```
int main() {
    #ifdef WIN32
        FILE* pipe = _popen(GNUPLOT_NAME, "w");
    #else
        FILE* pipe = popen(GNUPLOT_NAME, "w");
    #endif

    int size;
    cin >> size;

    double* t = new double[size];
    double* b = new double[size];

    for (int i = 0; i < size; i++) {
        cin >> t[i] >> b[i];
    }

    int polinomial_degree;
    cin >> polinomial_degree;

    Matrix A(size, polinomial_degree + 1);
    Matrix B(size, 1);

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < polinomial_degree + 1; j++) {
            A(i, j) = pow(t[i], j);
        }
        B(i, 0) = b[i];
    }

    cout << "A:\n" << A;

    Matrix A_transpose = A.transpose();

    cout << "A_T*A:\n" << A_transpose * A;

    Matrix A_inverse = (A_transpose * A).inverse();

    cout << "(A_T*A)^-1:\n" << A_inverse;

    Matrix A_transpose_B = A_transpose * B;

    cout << "A_T*b:\n" << A_transpose_B;

    Matrix x = A_inverse * A_transpose_B;

    cout << "x~:\n" << x;
```

```
    double min_t = t[0];
    double max_t = t[0];
    for (int i = 1; i < size; i++) {
        if (t[i] < min_t) {
            min_t = t[i];
        }
        if (t[i] > max_t) {
            max_t = t[i];
        }
    }

    fprintf(pipe, "set xrange [%lf:%lf]\n", min_t - 1, max_t + 1);

    double min_b = b[0];
    double max_b = b[0];

    for (int i = 1; i < size; i++) {
        if (b[i] < min_b) {
            min_b = b[i];
        }
        if (b[i] > max_b) {
            max_b = b[i];
        }
    }

    fprintf(pipe, "set yrange [%lf:%lf]\n", min_b - 1, max_b + 1);

    fprintf(pipe, "set grid\n");
    fprintf(pipe, "plot %lf*x**3 + %lf*x**2 + %lf*x**1 + %lf*x**0 , '-' using 1:2 with points\n",
        x(3, 0), x(2, 0), x(1, 0), x(0, 0));
    for (int i = 0; i < size; i++) {
        fprintf(pipe, "%f\t%f\n", t[i], b[i]);
    }
    fprintf(pipe, "e\n");
    fflush(pipe);


    #ifdef WIN32
        _pclose(pipe);
    #else
        pclose(pipe);
    #endif

    return 0;
}
```

Main function (without gnuplot):

```
int main() {
    int size;
    cin >> size;

    int* t = new int[size];
    int* b = new int[size];

    for (int i = 0; i < size; i++) {
        cin >> t[i] >> b[i];
    }

    int polinomial_degree;
    cin >> polinomial_degree;

    Matrix A(size, polinomial_degree + 1);
```

```
    Matrix B(size, 1);

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < polinomial_degree + 1; j++) {
            A(i, j) = pow(t[i], j);
        }
        B(i, 0) = b[i];
    }

    cout << "A:\n" << A;

    Matrix A_transpose = A.transpose();

    cout << "A_T*A:\n" << A_transpose * A;

    Matrix A_inverse = (A_transpose * A).inverse();

    cout << "(A_T*A)^-1:\n" << A_inverse;

    Matrix A_transpose_B = A_transpose * B;

    cout << "A_T*b:\n" << A_transpose_B;

    Matrix x = A_inverse * A_transpose_B;

    cout << "x~:\n" << x;

    return 0;
}
```

GitHub link: https://github.com/1kkiRen/AGLAJoint2