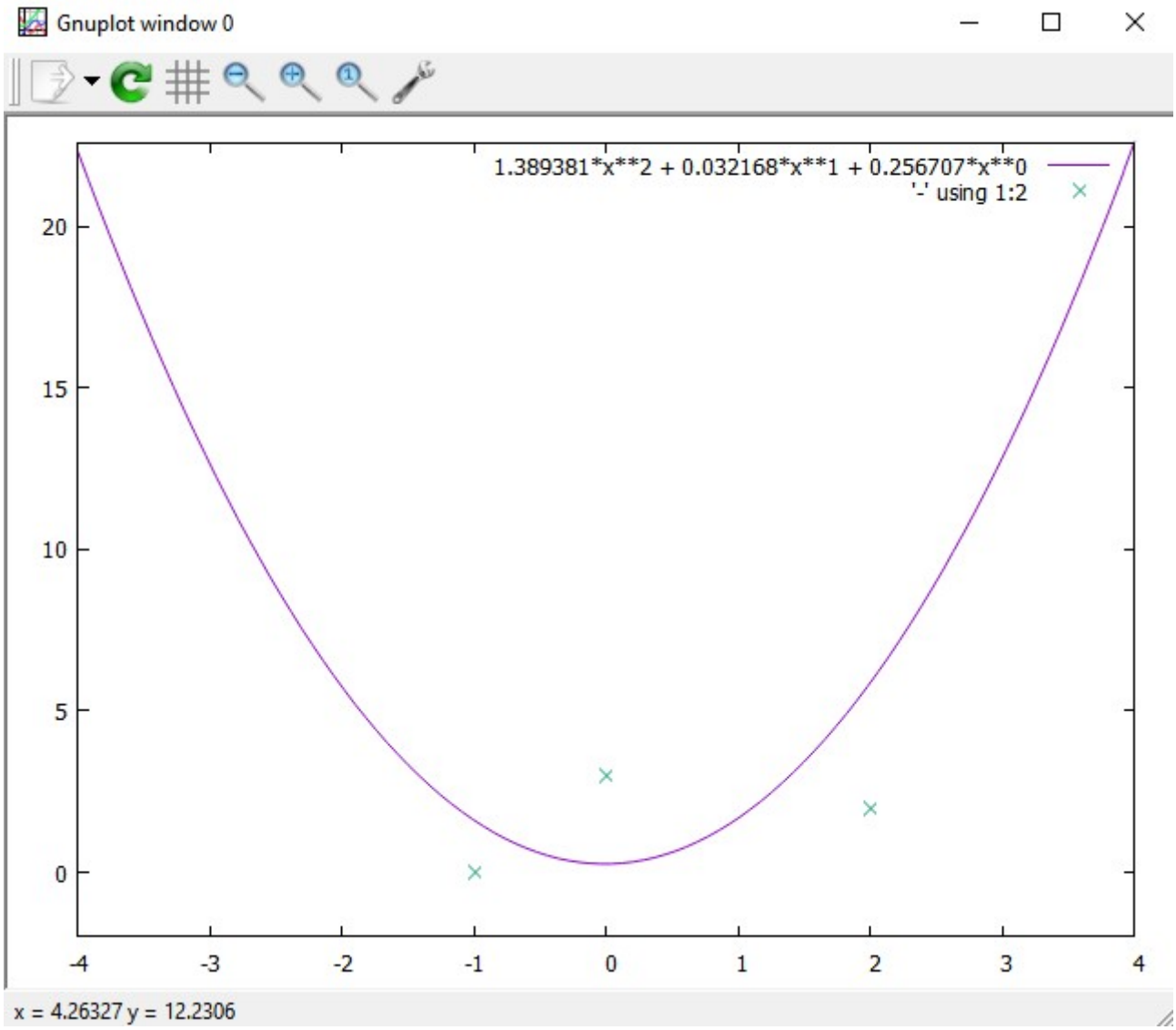
Analytical Geometry and Linear Algebra II

Dmitrii Kuzmin

April, 2023

DSAI-03
dm.kuzmin@innopolis.university



Experimental data:

t	b
0	3
2	2
-1	0

Main function:

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

    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_TA = A_transpose * A;

    Matrix A_inverse = inverse(A_TA);

    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;


    fprintf(pipe, "plot [-4 : 4] [-2 : 2] %lf*x**2 + %lf*x**1 + %lf*x**0 , '-' using 1:2 with
        points\n", x(0, 0), x(1, 0), x(2, 0));
    for (int i = 0; i < size; i++) {
        fprintf(pipe, "%d %d\n", t[i], b[i]);
    }
    fprintf(pipe, "e\n");
    fflush(pipe);


    #ifdef WIN32
        _pclose(pipe);
    #else
```

```
        pclose(pipe);
    #endif

    return 0;
}
```

GitHub link: https://github.com/1kkiRen/ALGAJoint2