```c
int matrix_invert(int dim, double** a, double** output) {
    if (a == NULL || output == NULL) {
        puts("matrix invert NULL ptr");
        return 1;
    }
    matrix_copy(dim, dim, a, output);
    double** i_matrix = create_identity(dim);
    for (int i = 0; i < dim; i++) {
        if (output[i][i] == 0) {
            int j;
            for (j = i; j < dim && output[j][i] == 0; j++);
            if (output[j][i] == 0) {
                return -1;
            }
            swap_row(i, j, output, output);
        }
        double scale = output[i][i];
        #pragma omp parallel for
        for (int j = 0; j < dim; j++) {
            output[i][j] = output[i][j] / scale;
            i_matrix[i][j] = i_matrix[i][j] / scale;
        }
        #pragma omp parallel for
        for (int j = i + 1; j < dim; j++) {
            double factor = output[j][i];
            #pragma omp parallel for
            for (int k = 0; k < dim; k++) {
                output[j][k] = output[j][k] - factor * output[i][k];
                i_matrix[j][k] = i_matrix[j][k] - factor * i_matrix[i][k];
            }
        }
    }
    for (int i = dim - 1; i > 0; i--) {
        #pragma parallel for
        for (int j = i - 1; j > -1; j--) {
            double factor = output[j][i];
            #pragma omp parallel for
            for (int k = 0; k < dim; k++) {
                output[j][k] = output[j][k] - factor * output[i][k];
                i_matrix[j][k] = i_matrix[j][k] - factor * i_matrix[i][k];
            }
        }
```