PA3 Part A Documentation

Program Description

Solves systems of equations given in the first 3 columns of the input files. I will accomplish this through inverse matrices, the dot product theorem, and my custom homogeneous system solver.

Important Library Details

- Eigen
  - Library path: the headers for the Eigen library are located in /usr/include/eigen3 on my Linux machine.
  - Library version: I have installed Eigen version 3.4.0.

Marginal Cases

- Invalid inputs:
  - System is underdetermined. System is inconsistent if $\vec{a}_1 \bullet \vec{a}_2 = \pm\ ||\vec{a}_1||\ ||\vec{a}_2||$ and $\vec{a}_1 \bullet \vec{b} = \pm\ ||\vec{a}_1||\ ||\vec{b}||$ where $A = \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \end{bmatrix}$. (derived from the dot product theorem, where θ = 0° or 180°)
  - System is inconsistent. System is inconsistent if $\vec{a}_1 \bullet \vec{a}_2 = \pm\ ||\vec{a}_1||\ ||\vec{a}_2||$ and $\vec{a}_1 \bullet \vec{b} \neq \pm\ ||\vec{a}_1||\ ||\vec{b}||$ where $A = \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \end{bmatrix}$. (derived from the dot product theorem, where θ = 0° or 180°)
  - The solution is trivial. Found with $det(A)\ \neq\ 0$.
- Invalid computations:
  - All important computations in the Eigen implementation methods were handled by Eigen, and the outputs have been checked.
  - All custom implementation methods simply added two doubles and stored them in the corresponding double element of a result matrix. The outputs have also been checked.

Design Choices

- $\vec{x}$ will be solved via the inverse matrix: $\vec{x} = A^{-1}\vec{b}$. This can be done because the invalid input cases for the first part of this section cover all cases where A is non-invertible.
- For solving the homogeneous systems, the homogeneous system solver will be used.

Pseudocode

```
Matrix(2,6) GetInputAsMatrix(const string &input_path)

Int main():
        CALL SolveFile() for each file

        RETURN 0

bool AreColinear(const vector &vec_1, const vector &vec_2):
        RETURN vec_1.dotProduct(vec_2) == vec_1.norm() * vec_2.norm() or
                        vec_1.dotProduct(vec_2) == vec_1.norm() * vec_2.norm() * -1

void SolveFile(const string &input_path, const string &output_path):
        DECLARE 2x6 matrix raw_input as CALL of GetInputAsMatrix with input_path
        OPEN output_file at output_path

        DECLARE matrix mat as the first two column vectors of raw_input
        DECLARE vector result as the third column vector of raw_input

        RUN PartA(mat, result, output_file)
        RUN PartB(mat, output_file)

        CLOSE output_file

void PartA(const matrix &mat, const vector &result, ofstream &output_file):
        IF IsColinear(mat.rowVector(0), mat.rowVector(1)):
                IF IsColinear(mat.rowVector(0), result):
                        PRINT “underdetermined” to output_file
                ELSE:
                        PRINT “inconsistent” to output_file
        ELSE:
                vector solution = mat.inverse() * result
                PRINT solution.transpose() to output_file

void PartB(const matrix &mat, ofstream &output_file):
        DECLARE vector solution

        DECLARE bool is_trivial as !SolveHomogeneousSystem(mat, solution)

        IF is_trivial:
                PRINT “trivial” to output_file
        ELSE:
                PRINT solution.transpose() to output_file
```