Matrix_Calculator

主要功能

输入模式

- 手动文字框输入
- 文件读取
- 输入流输入

矩阵功能

- 矩阵行列式
- 矩阵求逆
- 矩阵PLU分解
- 非方阵矩阵QR分解
- 矩阵SVD分解
- 矩阵Jordan标准型

代码框架

文件信息详见structure.txt,使用.pro文件组织项目,用QT框架构建UI

文件介绍

- core: 矩阵核心类
  - Matrix.h: 矩阵类
  - Entry.h: 有理数元素类
  - Polynomial.h: 有理数域多项式类
  - Matrix.cpp: 借助Entry类的计算实现线性计算，利用EntryPolynomial类进行特征值相关计算
  - Entry.cpp: 有理数元素计算
  - EntryPolynomial.cpp: 有理数域多项式计算及求解
- ui: 图形界面类
  - Mainwindow.h: 主窗口类
  - Mainwindow.cpp: 主窗口实现，从OperatorWidget类中获取矩阵操作，在Matrix类中进行计算，并将结果显示在BoardWidget类中
  - BoardWidget.h: 矩阵显示窗口类，使用QTableWidget显示矩阵
  - BoardWidget.cpp: 矩阵显示窗口实现，以及对于正交矩阵的归一化根号处理
  - OperatorWidget.h: 矩阵操作窗口类，提供矩阵操作
  - OperatorWidget.cpp: 矩阵操作窗口实现，输入输出矩阵及操作的按钮

- main.cpp: 主函数，构建主窗口
- Matrix_Calculator.pro: 项目文件，组织项目
- build文件夹: 编译生成的文件夹
- .vscode文件夹: VSCode配置文件夹
    - task.json: VSCode编译任务配置文件
    - launch.json: VSCode调试配置文件

功能位置

按钮界面

各种按钮的设置和操控 OperatorWidget.cpp

- explicit OperationWidget(QWidget *parent = nullptr);
- void showFunctionButtons();
- void hideFunctionButtons();
- void showInputButtons();
- void hideInputButtons();
- void hideStartButton();
- void showStartButton();
- void hideBackButton();
- void showBackButton();

开始按钮 MainWindow.cpp

- void handleStart();

主界面设置 MainWindow.cpp

- void setupUi();

返回按钮 MainWindow.cpp

- void handleBack();

矩阵输入

矩阵文件输入 MainWindow.cpp

- void handleFileInputMatrix();

矩阵输入框输入 MainWindow.cpp

- void handleBoxInputMatrix();

矩阵输入流输入 MainWindow.cpp

- void handleIostreamInputMatrix();

普通矩阵显示 BoardWidget.cpp

- void setMatrix(const Matrix &matrix);

左侧正交矩阵归一化 BoardWidget.cpp

- void setMatrixWithSquarerootLeft(const std::vector &norms,const Matrix &matrix);

右侧正交矩阵归一化 BoardWidget.cpp

- void setMatrixWithSquarerootRight(const Matrix &matrix,const std::vector& norms);

右侧正交矩阵转置归一化 BoardWidget.cpp

- void setTransposeMatrixWithSquarerootRight(const Matrix &matrix,const std::vector& norms);

奇异值显示 BoardWidget.cpp

- void setMatrixWithSVD(const Matrix &matrix);

矩阵计算

基础线性代数功能 Matrix.cpp

- int getRows() const;
- int getCols() const;
- std::vector getRow(int row) const;
- std::vector getColumn(int col) const;
- Entry getEntry(int row, int col) const;
- void setEntry(int row, int col, const Entry &value);
- void setRow(int row, const std::vector &values);
- void setColumn(int col, const std::vector &values);
- void rowPopBack();
- void columnPopBack();
- bool hasValue(int row, int col) const;
- void reduceAll();
- Matrix getIdentityMatrix(int n) const;
- Matrix getMatrixTranspose() const;
- Matrix LeftMultiply(const Matrix &other) const;
- std::vector LeftMultiplyVector(const std::vector &other) const;
- Matrix RightMultiply(const Matrix &other) const;
- std::vector RightMultiplyVector(const std::vector &other) const;

- Entry norm(const std::vector &a);
- bool isZeroVector(const std::vector &a);
- std::vector entryMultiplyOnVector(const Entry &coefficient, const std::vector &a) const;
- Entry getInnerProduct(const std::vector &a, const std::vector &b);
- std::pairstd::vector, std::vector> getColumnNormsAndInverse() const;
- int getRank() const;

矩阵特征值及相关计算 Matrix.cpp

- EntryPolynomial calculateDeterminant(const std::vectorstd::vector> &polyMatrix) const;
- EntryPolynomial getCharacteristicPolynomial() const;
- std::vector<std::pair<Entry, int>> getEigenvalues() const;
- Matrix getOrthogonalEigenBasis(const std::vector<std::pair<Entry, int> >&eigenvalues) const;

矩阵行列式计算 Matrix.cpp

- Entry getDeterminant() const;

矩阵求逆 Matrix.cpp

- Matrix inverse() const;

矩阵PLU分解 Matrix.cpp

- std::tuple<Matrix, Matrix, std::vector, std::vector> pluDecomposition() const;

非方阵矩阵QR分解 Matrix.cpp

- std::pair<Matrix, Matrix> qrDecomposition() const;

矩阵SVD分解 Matrix.cpp

- std::tuple<Matrix, Matrix, Matrix> getSVDdecomposition() const;

矩阵Jordan标准型 Matrix.cpp

- Matrix getJordanForm() const;

窗口控制

计算窗口控制 MainWindow.cpp

- void handleLuDecomposition();
- void handleInverse();
- void handleDeterminant();
- void handleQrDecomposition();
- void handleSvdDecomposition();
- void handleJordanForm();

功能思路和算法介绍

矩阵输入

手动文字框输入：

- 使用QInputDialog输入矩阵
- 用QMessageBox进行输入错误处理
- 输错了可以允许重新输入
    - esc可以退出，也有防止误触功能，会弹出QMessageBox确认

文件读取：

- 使用QFileDialog弹出文件选取框
- 用QMessageBox进行文件打开错误处理
- 文件格式错误会弹出QMessageBox
- 错误之后自动返回

输入流输入：

- 使用cin输入矩阵
- 用cout进行输入错误处理
- 输错了可以允许重新输入
- 输错了也可以选择退出
- 受到iostream的限制，输入中途无法点击返回

矩阵计算

矩阵行列式计算

- 借助基本的行列式展开递归方法

矩阵求逆

- 利用行列式计算和伴随矩阵（也可以理解成Cramer法则）

矩阵PLU分解

- 借助高斯消元法

非方阵矩阵QR分解

- 借助Givens变换
- Givens变换有效的解决了QR分解中不满秩矩阵的分解问题，同时给不满秩的部分创造了有理数的正交基，比Gram-Schmidt方法在这点上占优
- 不优先做正交化，避免根式运算，最后做归一化，全程使用有理数运算（包括Givens矩阵）

矩阵SVD分解

- 特征多项式的处理及其必要性
  - 矩阵的SVD分解需要对ATA和AAT两个矩阵做特征值分解
  - 为了有理数保留其精确性，传统的迭代方法不可使用
  - 行列变换算法一定不能存在因为这个问题等价与任意次方程的求根公式问题
  - 通过定义有理数向量EntryPolynomial，我们可以对$\mathbb{Q}[x]$ 上的矩阵进行运算
  - 通过类似的递归子式展开计算，我们可以计算出特征多项式
- 特征多项式的求根
  - 有理数域多项式的求根可以先通分，其根的分子必整除常数项，分母必整除首项
  - 优先处理数论性质特殊而且易于观察的根0
  - 我们对首项和常数项进行质因数分解，并且用一个p_i, α_i的数对向量记录结果
  - 不断使用nextFactor函数生成新的试根，用plugIn函数验证根
  - 使用divideLinearFactor函数将特征多项式除以线性因子，得到新的商式
  - 每得到一个根后，除尽为止，记录根的代数重数，最后返回一个λ_i, d_i的数对向量作为其在$[Q]$的分解
- 特征向量的计算
  - QR分解对不可逆矩阵来说是不唯一的，Givens变换给出的QR分解中，不能明确的指出R中那些列是否与前面的列线性相关。
  - 这个时候不可避免的使用Gram-Schmidt正交化方法，但这个时候我们恰恰不需要那些非主元对应的向量。
  - 我们用G-S正交化原矩阵，并在R中记录其系数，对于全零列（非主元）跳过，避免出现除0风险，最终得到一个高的列缺少矩阵Q一个行缺少的长矩阵R，实际意义是只保留了有效向量的信息和对应系数。一并返回的还有Q的列向量在原矩阵中的位置，作为特征向量的索引。
  - $Q'R'x = 0$转化为$R'x = 0$,其中的解的基可以用主元列上三角的部分生成其余列的系数构造，并在进入正交基的时候做G-S正交化，最终得到特征向量。
- 特征向量的归一化
  - 为了计算简便，之前的一切处理依旧是用了有理数的未归一化特征向量
  - 这里我们利用BoardWidget的归一化显示功能，对特征向量做归一化并显示

矩阵Jordan标准型

- Jordan标准型的主要计算方法是计算$\mathrm{Rank}\ (A - \lambda I)^k$，从而解决代数重数不等于几何重数的方法
- getRank函数，利用G-S正交化类似的方法可以得到Rank，由于不要求对应的可逆矩阵，所以不需要记录系数
- 根据计算出来的rank来累进到Jordan块的阶数，最终得到Jordan标准型

创新探索

附加功能

- 在标准要求之外的输入框输入，以及实时的可视化
- 对错误输入的反馈和重新输入及退出选择
- 加入等号的显示，整体显示更为清晰

代码安全

- 内存安全
  - 使用shared_ptr管理矩阵，减少内存泄漏风险
  - 对于new出来的部分及时delete
- 数据异常处理及风险排查
  - 对于分母为0的数据抛出异常
  - 对于行列不符计算要求的矩阵抛出异常
  - 对于特征值不符合有理数要求的矩阵抛出异常，避免死循环
  - 对于多项式运算及时消去0的高次系数项
  - 对于对角化矩阵的操作，预留了给非对称矩阵的操作，也预留了不可对角化的异常
  - 检测求根时只存在常数项的死循环风险，抛出异常
- 数据规模溢出风险
  - 对有理数及时约分，减少数据规模
  - 使用long long作为有理数的分子分母处理
  - 对于超出long long范围的计算过程整数抛出异常

效率提升

- 在素数计算过程中，先筛法计算小规模素数，再进行大规模计算
- 单独获取getRank函数，避免重复计算
- EntryPolynomial.cpp 124行，先把第一次的根除去，减少一次代入
- Matrix.cpp 766行，先把第一次累计几何重数给出，避免后面再次计算rank
- (to be done) 上三角矩阵求解用追赶法比直接求逆更快

！！！特别提示！！！

- 输入流输入中途无法点击返回，请注意
- 输入流输入需要尽快完成，否则会卡住Qt的进程
- 本次矩阵涉及有理数，并且为了保证精确值，加减乘除都不可避免造成大分子大分母，虽然使用long long作为分子分母处理，但依旧可能溢出。其中最坏的情况应该是$O\left(p^{n^2}\right)$量级的，其中p是数字的规模，n是矩阵的阶数。不建议使用过大的阶数和数字。