# Explanations of Web Tutorial 1

## 1. My Information

- Name: Qiaoru Li
- School ID: 20221939
- Class: International Class of Software Engineering

## 2. Files and Explanations

**Note: Because of the javascript-related functionality used in the project, it is not feasible to run individual HTML documents separately. Please compile them through a local server!**

The project has 2 pages in total, which are Main Page and Project Page. The explanations of key files or folders are the following:

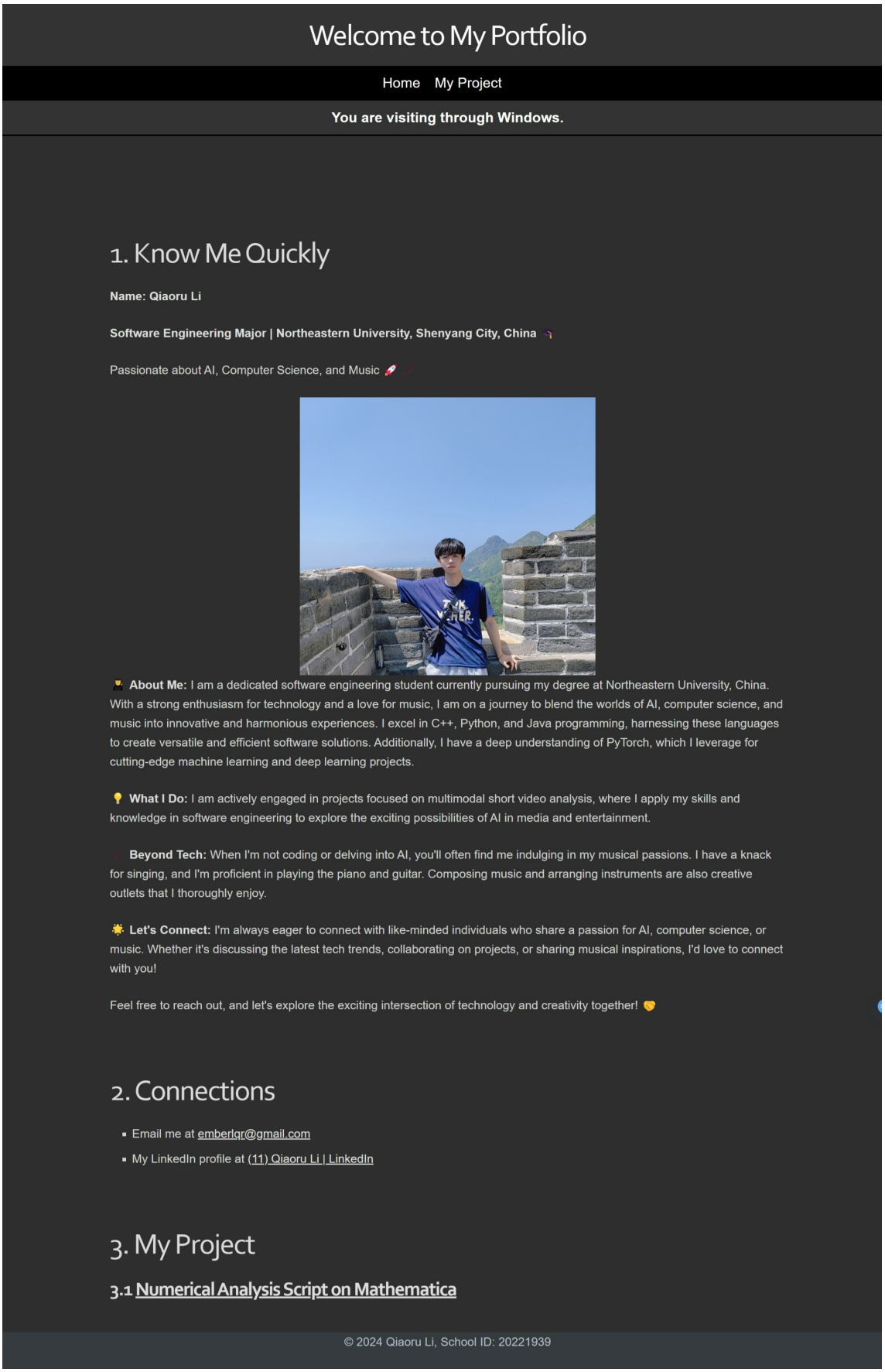
- `index.html` is the main page.
- `MainpageContent.html` is the main body of `index.html`.
- `nav.html` is the navigation which can be reused in all pages.
- `project.html` is the my-project introduction page.
- `ProjectContent.html` is the main body of `project.html`.
- `styles.css` defines the styles of `nav.html`.
- `/images` directory stores necessary image files used in the HTML files.
- `Main Page.jpg` is the screenshot of the main page.
- `Project Page.jpg` is the screenshot of the project page.

In the main body (`MainpageContent.html` and `ProjectContent.html`, I used markdown style, which is based on Typora's markdown framework.

Specially, I added a module that detects the user's system and shows what system the user is using to access the page. (shown in the screenshots)

## 3. Project screenshots

- > Main Page.jpg




- Project Page.jpg

# Numerical Analysis Script

Github Link: EmberQR/Numerical-Analysis: A Mathematica script used to calculate some common operations of numerical analysis. (github.com)

## 1. Introduction

This project provides a collection of numerical analysis scripts designed for use with **Mathematica**. These scripts are useful for performing various numerical computations commonly encountered in mathematical and engineering problems.

- **Effective Digits Calculation**: Determine the number of significant digits in which an approximate value $a$ approximates the exact value $b$ using `EffectiveDigits[a, b]`.
- **Gauss Elimination Operation Count**: Calculate the number of multiplication and division operations in sequential Gauss elimination for a system with $n$ unknowns using `GaussN[n]`.
- **LU Decomposition**: Perform LU decomposition on matrix $M$ with `LUDe[M]`, returning {L, U} where $L$ is a unit lower triangular matrix and $U$ is an upper triangular matrix.
- **Cholesky Decomposition**: Decompose matrix $M$ into a lower triangular matrix $G$ and its transpose $G^T$ such that $M = GG^T$ using `ChDe[M]`, returning $G$.
- **Positive Definite Matrix Check**: Determine if matrix $A$ is positive definite with `PositiveDefiniteMatrixQ[A]`.
- **Principal Minors Calculation**: Compute the principal minors of matrix $A$ using `PrincipalMinors[A]`.
- **Crout Decomposition for Tridiagonal Matrices**: Perform Crout decomposition on tridiagonal matrix $A$ with `CroutDe[A]`, returning a lower triangular matrix and a unit upper triangular matrix (more efficient for tridiagonal matrices).
- **General Crout Decomposition**: Perform Crout decomposition on any matrix $A$ using `CroutDeCo[A]`, returning a lower triangular matrix and a unit upper triangular matrix.
- **Norm Calculation**: Compute the norm of a vector or matrix $A$ with `Norm[A, p]`, where $p$ can be 1, 2, Infinity, or "Frobenius" (Frobenius norm).
- **Spectral Radius Calculation**: Determine the spectral radius of matrix $A$ using `PBJ[A]`.
- **Condition Number Calculation**: Calculate the $p$-condition number of matrix $A$ with `Cond[A, p]`.
- **Jacobi Iteration Matrix**: Compute the iteration matrix for the Jacobi method using `Jacobi[A]`.
- **Gauss-Seidel Iteration Matrix**: Compute the iteration matrix for the Gauss-Seidel method using `GS[A]`.
- **Strict Diagonal Dominance Check**: Determine if matrix $A$ is strictly diagonally dominant with `IsStrictlyDiagonallyDominant[A]`.
- **SOR Method Iteration Matrix**: Calculate the iteration matrix for the SOR method with `SOR[A, w]`, where $w$ is the relaxation factor.
- **QR Decomposition**: Perform QR decomposition on matrix $M$ using `QRDecomposition[M]`, returning an orthogonal matrix and an upper triangular matrix.

## 2. Screenshots

- `CroutDe` Function:

```
In[•]:= A = {{2, -1, 0, 0}, {-1, 2, -1, 0}, {0, -1, 2, -1}, {0, 0, -1, 2}};
        CroutDe[A]
```

$$Out[\bullet]= \left\{\begin{pmatrix} 2 & 0 & 0 & 0 \\ -1 & \frac{3}{2} & 0 & 0 \\ 0 & -1 & \frac{4}{3} & 0 \\ 0 & 0 & -1 & \frac{5}{4} \end{pmatrix}, \begin{pmatrix} 1 & -\frac{1}{2} & 0 & 0 \\ 0 & 1 & -\frac{2}{3} & 0 \\ 0 & 0 & 1 & -\frac{3}{4} \\ 0 & 0 & 0 & 1 \end{pmatrix}\right\}$$

- `LUDe` Function:

```
A = {{1, 5, -3}, {-2, -7, 3}, {4, 9, 6}};
LUDe[A]
```

$$Out[\bullet]= \left\{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & -\frac{11}{\cdot} & 1 \end{pmatrix}, \begin{pmatrix} 1 & 5 & -3 \\ 0 & 3 & -3 \\ 0 & 0 & 7 \end{pmatrix}\right\}$$

- SOR Function:

```
M = {{4, -2, -4}, {-2, 17, 10}, {-4, 10, 9}};
w = 1.25; (*松弛因子*)
SOR[M, w]
```

Out[•]//MatrixForm=

$$\begin{pmatrix} -0.25 & 0.625 & 1.25 \\ -0.0367647 & -0.158088 & -0.551471 \\ -0.0878268 & 0.566789 & 1.21038 \end{pmatrix}$$