Solution to the 2nd Homework

Yimeng Zhu

November 16, 2019

Contents

1 Eigen 矩阵运算 (3 分，约 2 小时)

Eigen(http://eigen.tuxfamily.org) 是常用的 C++ 矩阵运算库，具有很高的运算效率。大部分需要在 C++ 中使用矩阵运算的库，都会选用 Eigen 作为基本代数库，例如 Google Tensorflow，Google Ceres，GTSAM 等。本次习题，你需要使用 Eigen 库，编写程序，求解一个线性方程组。为此，你需要先了解一些有关线性方程组数值解法的原理。设线性方程 Ax = b，在 A 为方阵的前提下，请回答以下问题:

1. 在什么条件下，x 有解且唯一?
2. 高斯消元法的原理是什么?
3. QR 分解的原理是什么?
4. Cholesky 分解的原理是什么?
5. 编程实现 A 为 100 × 100 随机矩阵时，用 QR 和 Cholesky 分解求 x 的程序。你可以参考本次课用到的 useEigen 例程。提示: 你可能需要参考相关的数学书籍或文章。请善用搜索引擎。Eigen 固定大小矩阵最大支持到 50，所以你会用到动态大小的矩阵。

Solution:

1. $rank(A) = rank(A|b) = dim(x) =$ number of variables
2. - Step 1: Using following 3 elementary row operations to modify the matrix until the lower left corner is filled with 0, as much as possible (also called row echelon form):
     - swapping two rows
     - multiplying a row by a nonzero number
     - adding a multiple of one row to another row.
   - Step 2: Using back substitution to find the solution of the equations, i.e. using last equation to solve x_n, and substitute it into the previous equation to solve x_{n-1}, and repeating through x_1.
3. - If A is a square matrix, A can be decomposed as $A = QR$, where where Q is an orthogonal matrix and R is an upper triangular matrix.
   - More generally, we can factor a complex m×n matrix A, with $m \geq n$, A can be decomposed as

     $$A = QR = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = [Q_1, Q_2] \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1 R_1$$

     where R_1 is an n×n upper triangular matrix, 0 is an (m − n)×n zero matrix, Q_1 is m×n, Q_2 is m×(m − n), and Q_1 and Q_2 both have orthogonal columns.

   After decomposition, the equation $Ax = b$ can be solved more numerically stable, even for the under and over determined case.
4. If A is a symmetric positive definite matrix, A can be decomposed as $A = LL^T$, where L is a lower triangular matrix. Then, the equation $Ax = b$ can be efficiently solved by a forward substitute and a backward substitute, as it can be write as $LL^T x = b$. The resolution can be done in following 2 steps:
   (a) resolve the equation $Ly = b$ with forward substitute.
   (b) resolve the equation $L^T x = y$ with backward substitute.
5. see folder 1.5.

2 几何运算练习 (2 分，约 1 小时)

下面我们来练习如何使用 Eigen/Geometry 计算一个具体的例子。

设有小萝卜一号和小萝卜二号位于世界坐标系中。小萝卜一号的位姿为:$q_1 = [0.55, 0.3, 0.2, 0.2], t_1 = [0.7, 1.1, 0.2]^T$ (q 的第一项为实部)。这里的 q 和 t 表达的是 Tcw，也就是世界到相机的变换关系。小萝卜二号的位姿为 $q_2 = [-0.1, 0.3, -0.7, 0.2], t2 = [-0.1, 0.4, 0.8]^T$。现在，小萝卜一号看到某个点在自身的坐标系下，坐标为 $p_1 = [0.5, -0.1, 0.2]^T$，求该向量在小萝卜二号坐标系下的坐标。请编程实现此事，并提交你的程序。

提示:

1. 四元数在使用前需要归一化。
2. 请注意 Eigen 在使用四元数时的虚部和实部顺序。
3. 参考答案为 $p_2 = [1.08228, 0.663509, 0.686957]^T$。你可以用它验证程序是否正确。

Solution: see folder 2.

3 旋转的表达 (2 分，约 1 小时)

课程中提到了旋转可以用旋转矩阵、旋转向量与四元数表达，其中旋转矩阵与四元数是日常应用中常见的表达方式。请根据课件知识，完成下述内容的证明。

1. 设有旋转矩阵 R，证明 $R^T R = I$ 且 $detR = +1$。
2. 设有四元数 q，我们把虚部记为 ϵ，实部记为 η，那么 $q = (\epsilon, \eta)$。请说明 ϵ 和 η 的维度。
3. 定义运算 $+$ 和 $\oplus$ 为:

$$q^+ = \begin{bmatrix} \eta \mathbf{I} + \epsilon^\times & \epsilon \\ -\epsilon^T & \eta \end{bmatrix}, \quad q^\oplus = \begin{bmatrix} \eta \mathbf{I} - \epsilon^\times & \epsilon \\ -\epsilon^T & \eta \end{bmatrix}, \tag{1}$$

其中运算 $\times$ 含义与 $\hat{}$ 相同，即取 ϵ 的反对称矩阵 (它们都成叉积的矩阵运算形式)，$\mathbf{1}$ 为单位矩阵。请证明对任意单位四元数 q_1, q_2，四元数乘法可写成矩阵乘法:

$$q_1 q_2 = q_1^+ q_2 \tag{2}$$

或者

$$q_1 q_2 = q_2^\oplus q_1 \tag{3}$$

Solution:

1. **Proof:** As in lecture, coordinate system (e_1, e_2, e_3) is rotated to (e_1', e_2', e_3'), for the point A, with original coordinate (a_1, a_2, a_3) and rotated coordinate (a_1', a_2', a_3'), we have:

$$\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} e_1' & e_2' & e_3' \end{bmatrix} \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix}$$

Multiply $\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}^T$ on both sides from left, we have:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} e_1^T e_1' & e_1^T e_2' & e_1^T e_3' \\ e_2^T e_1' & e_2^T e_2' & e_2^T e_3' \\ e_3^T e_1' & e_3^T e_2' & e_3^T e_3' \end{bmatrix} \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} = R \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix}$$

Multiply $\begin{bmatrix} e_1' & e_2' & e_3' \end{bmatrix}^T$ on both sides from left, we have:

$$\begin{bmatrix} e_1'^T e_1 & e_1'^T e_2 & e_1'^T e_3 \\ e_2'^T e_1 & e_2'^T e_2 & e_2'^T e_3 \\ e_3'^T e_1 & e_3'^T e_2 & e_3'^T e_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = R^T \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix}$$

Thus, we have:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = R \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} = RR^T \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Therefore, $RR^T = I$.

2. $dim(\epsilon) = 3, dim(\eta) = 1$

3. Let $q_1 = \begin{bmatrix} \epsilon_1 \\ \eta_1 \end{bmatrix}$, $q_2 = \begin{bmatrix} \epsilon_2 \\ \eta_2 \end{bmatrix}$, according to the formula in the lecture,

$$
\begin{aligned}
q_1 q_2 &= \begin{bmatrix} \eta_1 \epsilon_2 + \eta_2 \epsilon_1 + \epsilon_1 \times \epsilon_2 \\ \eta_1 \eta_2 - \epsilon_1^T \epsilon_2 \end{bmatrix} &= \begin{bmatrix} \eta_{\mathbf{1}} \mathbf{I} + \epsilon_{\mathbf{1}}{}^{\times} & \epsilon_{\mathbf{1}} \\ -\epsilon_{\mathbf{1}}{}^{T} & \eta_{\mathbf{1}} \end{bmatrix} \begin{bmatrix} \epsilon_2 \\ \eta_2 \end{bmatrix} = q_1^{+} q_2 \\
&= \begin{bmatrix} \eta_1 \epsilon_2 + \eta_2 \epsilon_1 - \epsilon_2 \times \epsilon_1 \\ \eta_1 \eta_2 - \epsilon_1^T \epsilon_2 \end{bmatrix} &= \begin{bmatrix} \eta_{\mathbf{2}} \mathbf{I} - \epsilon_{\mathbf{2}}{}^{\times} & \epsilon_{\mathbf{2}} \\ -\epsilon_{\mathbf{2}}{}^{T} & \eta_{\mathbf{2}} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \eta_1 \end{bmatrix} = q_2^{\oplus} q_1
\end{aligned}
$$

4 罗德里格斯公式的证明 (2 分，约 1 小时)

罗德里格斯公式描述了从旋转向量到旋转矩阵的转换关系。设旋转向量长度为 θ，方向为 $\mathbf{k}$，那么旋转矩阵 $\mathbf{R}$ 为:

$$\mathbf{R} = \cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{k}\mathbf{k}^T + \sin\theta\mathbf{k}\hat{} \tag{4}$$

1. 我们在课程中仅指出了该式成立，但没有给出证明。请你证明此式。

   提示：参考https://en.wikipedia.org/wiki/Rodrigues_rotation_formula。

2. 请使用此式请明 $\mathbf{R}^{-1} = \mathbf{R}^T$ 。

Solution:

1. As shown in figure 1, the vector $\mathbf{v}$ is rotated by angle θ along the unit vector $\mathbf{v}$, which defines the rotation axis. Furthermore, the $\mathbf{v}$ can be decomposed into components parallel and perpendicular to the axis $\mathbf{k}$ as:

$$\mathbf{v} = \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$$

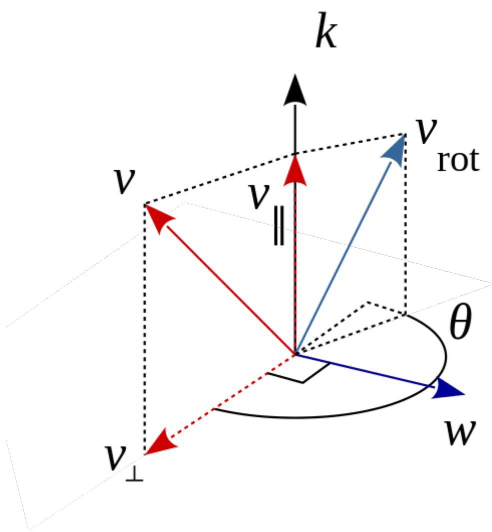

Figure 1: Rodrigues' rotation formula

The vector parallel and perpendicular to the axis $\mathbf{k}$ can be written correspondingly as:

$$\mathbf{v}_{\parallel} = (\mathbf{v}\cdot\mathbf{k})\mathbf{k}$$
$$\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{\parallel} = \mathbf{v} - (\mathbf{v}\cdot\mathbf{k})\mathbf{k} = -\mathbf{k}\times(\mathbf{k}\times\mathbf{v})$$

The rotation does not change the parallel component. For the perpendicular component, only the direction is rotated by angle θ while the magnitude remains the same, i.e,

$$\mathbf{v}_{\mathbf{rot}\parallel} = \mathbf{v}_{\parallel}$$
$$\mathbf{v}_{\mathbf{rot}\perp} = \cos\theta\mathbf{v}_{\perp} + \sin\theta\mathbf{k}\times\mathbf{v}_{\perp}$$

Since $\mathbf{k}\times\mathbf{v}_{\perp} = \mathbf{k}\times\mathbf{v}$, the above equation can be rewritten as:

$$\mathbf{v_{rot\perp}} = \cos\theta \mathbf{v}_\perp + \sin\theta \mathbf{k} \times \mathbf{v}$$

Thus, the rotated vector $\mathbf{v_{rot}}$ can be represented as sum of parallel and perpendicular components:

$$\begin{aligned}
\mathbf{v_{rot}} &= \mathbf{v_{rot\|}} + \mathbf{v_{rot\perp}} \\
&= \mathbf{v}_\| + \cos\theta \mathbf{v}_\perp + \sin\theta \mathbf{k} \times \mathbf{v} \\
&= \mathbf{v}_\| + \cos\theta(\mathbf{v} - \mathbf{v}_\|) + \sin\theta \mathbf{k} \times \mathbf{v} \\
&= \cos\theta \mathbf{v} + (\mathbf{1} - \cos\theta)\mathbf{v}_\| + \sin\theta \mathbf{k} \times \mathbf{v} \\
&= \cos\theta \mathbf{v} + (\mathbf{1} - \cos\theta)(\mathbf{k} \cdot \mathbf{v})\mathbf{k} + \sin\theta \mathbf{k} \times \mathbf{v} \\
&= (\cos\theta \mathbf{I} + (\mathbf{1} - \cos\theta)\mathbf{k} \cdot \mathbf{k^T} + \sin\theta \widehat{\mathbf{k}})\mathbf{v} \qquad = \mathbf{Rv}
\end{aligned}$$

Thus, we have

$$\mathbf{R} = \cos\theta \mathbf{I} + (\mathbf{1} - \cos\theta)\mathbf{k} \cdot \mathbf{k^T} + \sin\theta \widehat{\mathbf{k}}$$

2. As $\mathbf{R^{-1}}$ can be interpreted as inverse operation of $\mathbf{R}$, i.e. rotate a vector by angle $-\theta$ around the same axis $\mathbf{k}$, one can write $\mathbf{R^{-1}}$ as:

$$\mathbf{R^{-1}} = \cos(-\theta)\mathbf{I} + (\mathbf{1} - \cos(-\theta))\mathbf{kk^T} + \sin(-\theta)\widehat{\mathbf{k}} = \cos\theta \mathbf{I} + (\mathbf{1} - \cos\theta)\mathbf{kk^T} - \sin\theta \widehat{\mathbf{k}}$$

let $\mathbf{k} = (k_1, k_2, k_3)$, the $\widehat{\mathbf{k}}$ is defined as:

$$\widehat{\mathbf{k}} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix}$$

Therefore, the last term of R^{-1} can be written as:

$$-\sin\theta \widehat{\mathbf{k}} = -\sin\theta \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} = \sin\theta \begin{bmatrix} 0 & k_3 & -k_2 \\ -k_3 & 0 & k_1 \\ k_2 & -k_1 & 0 \end{bmatrix} = \sin\theta \widehat{\mathbf{k}}^T$$

Thus, the $\mathbf{R^T}$ can be written as:

$$\begin{aligned}
R^T &= (\cos\theta \mathbf{I} + (\mathbf{1} - \cos\theta)\mathbf{k} \cdot \mathbf{k^T} + \sin\theta \widehat{\mathbf{k}})^T \\
&= (\cos\theta \mathbf{I})^{\mathbf{T}} + ((\mathbf{1} - \cos\theta)\mathbf{k} \cdot \mathbf{k^T})^{\mathbf{T}} + (\sin\theta \widehat{\mathbf{k}})^T \\
&= \cos\theta \mathbf{I} + (\mathbf{1} - \cos\theta)\mathbf{kk^T} - \sin\theta \widehat{\mathbf{k}} \\
&= \mathbf{R^{-1}}
\end{aligned}$$

5 四元数运算性质的验证 (1 分，约 1 小时)

课程中介绍了单位四元数可以表达旋转。其中，在谈论用四元数 $\mathbf{q}$ 旋转点 $\mathbf{q}$ 时，结果为:

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-\mathbf{1}} \tag{5}$$

我们说，此时 $\mathbf{p}'$ 必定为虚四元数 (实部为零)。请你验证上述说法。此外，上式亦可写成矩阵运算:$\mathbf{p} = \mathbf{Q}\mathbf{p}$。请根据你的推导，给出矩阵 $\mathbf{Q}$。注意此时 $\mathbf{p}$ 和 $\mathbf{p}'$ 都是四元数形式的变量，所以 $\mathbf{Q}$ 为 4×4 的矩阵。提示: 如果使用第 3 题结果，那么有:

$$\begin{aligned} \mathbf{p}' &= \mathbf{q}\mathbf{p}\mathbf{q}^{-\mathbf{1}} = \mathbf{q}^{+}\mathbf{q}^{+}\mathbf{q}^{-\mathbf{1}} \\ &= \mathbf{q}^{+}\mathbf{q}^{-\mathbf{1}^{\oplus}}\mathbf{p} \end{aligned} \tag{6}$$

从而可以导出四元数至旋转矩阵的转换方式:

$$\mathbf{R} = Im(\mathbf{q}^{+}\mathbf{q}^{-\mathbf{1}^{\oplus}}). \tag{7}$$

其中 Im 指取出虚部的内容。

Solution: let $p = (v, 0)$ and $q = (\epsilon, \eta)$ as in lecture and question 3, according to the formula about quaternion one can have:

$$qp = (\eta v + \epsilon \times v, -\epsilon^T v)$$

$$q^{-1} = (-\epsilon, \eta)$$

and

$$\begin{aligned} Re\{q^{-1}pq\} &= (-\epsilon^T v)\eta - (\eta v + \eta \times v)^T \epsilon \\ &= -\epsilon^T v\eta - \eta v^T \epsilon + (\epsilon \times v)^T \epsilon \\ &= 0 \end{aligned}$$

$$\begin{aligned} Im\{q^{-1}pq\} &= -\epsilon^T v(-\epsilon) + \eta(\eta v + \epsilon \times v) + (\eta v + \epsilon \times v) \times (-\epsilon) \\ &= \epsilon^T \epsilon v + \eta\eta v + \eta\hat{\epsilon}v + \hat{\epsilon}\eta v + \hat{\epsilon}\hat{\epsilon}v \\ &= (\epsilon^T \epsilon + \eta^2 + 2\eta\hat{\epsilon} + \hat{\epsilon}\hat{\epsilon})v \end{aligned}$$

Thus,

$$R = \begin{bmatrix} 0 & 0_{1\times 3} \\ 0_{3\times 1} & \epsilon^T \epsilon + \eta^2 + 2\eta\hat{\epsilon} + \hat{\epsilon}\hat{\epsilon} \end{bmatrix}$$

6 * 熟悉 C++11 (2 分，约 1 小时)

请注意本题为附加题。C++ 是一门古老的语言，但它的标准至今仍在不断发展。在 2011 年、2014 年和 2017 年，C++ 的标准又进行了更新，被称为 C++11，C++14，C++17。其中，C++11 标准是最重要的一次更新，让 C++ 发生了重要的改变，也使得近年来的 C++ 程序与你在课本上 (比如谭浩强) 学到的 C++ 程序有很大的不同。你甚至会惊叹这是一种全新的语言。C++14 和 C++17 则是对 11 标准的完善与扩充。

越来越多的程序开始使用 11 标准，它也会让你在写程序时更加得心应手。本题中，你将学习一些 11 标准下的新语法。请参考本次作业 books/目录下的两个 pdf，并回答下面的问题。

设有类 A，并有 A 类的一组对象，组成了一个 vector。现在希望对这个 vector 进行排序，但排序的方式由 A.index 成员大小定义。那么，在 C++11 的语法下，程序写成:

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  class A {
8  public:
9      A(const int& i ) : index(i) {}
10     int index = 0;
11 };
12
13 int main() {
14     A a1(3), a2(5), a3(9);
15     vector<A> avec{a1, a2, a3};
16     std::sort(avec.begin(), avec.end(), [](const A&a1, const A&a2) {return a1.index<a2.index
       ;});
17     for ( auto& a: avec ) cout<<a.index<<" ";
18     cout<<endl;
19     return 0;
20 }
```

请说明该程序中哪些地方用到了 C++11 标准的内容。提示: 请关注范围 for 循环、自动类型推导、lambda 表达式等内容。

Solution:

- range based for loop: line 17

  ```
  for ( auto& a: avec )
  ```

- auto type inference: line 17

  ```
  auto& a: avec
  ```

- lambda function: line 16

  ```
  [](const A&a1, const A&a2) {return a1.index<a2.index;}
  ```