# 2 熟悉 Eigen 矩阵运算 (3 分,约 2 小时)

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?

R(A)=n，n为未知数个数。

即矩阵的秩等于未知数个数。

## 2. 高斯消元法的原理是什么?

高斯消元是逐步的消除线性方程组中的未知数，来构造一个行阶梯矩阵（上/下三角），然后回代解出每一个未知数的办法。举例来说,有三个等式

构造增广矩阵

这个矩阵得到(消除一个未知数)

再得到上三角矩阵（再消除一个未知数）

也就得到了三个等式

从下往上代入解出的未知数，得到z=1,y=2,x=3。

## 3. QR 分解的原理是什么?

QR分解是把一个满秩矩阵A分解成A=QR的式子。其中Q是一个正交矩阵，R是一个上三角矩阵。这样Ax=b就可以转化为QRx=b，左右代入得到

Q很容易计算，R是上三角矩阵，最后就跟上题高斯消元一样回代解出结果。

其实它是把A矩阵（举例3x3矩阵）的列向量组进行gram-schmidt正交化，得到一组正交的基，A矩阵等于一组常数和正交基组的乘积，比如

即

我们知道A，我们知道q，则R可知。

这种分解可以简化计算，主要是避免算.

## 4. Cholesky 分解的原理是什么?

同样有Ax=B，如果A对称正定，则存在，L为对角元为正数的下三角矩阵。（Cholesky分解即是LU分解的特殊情况）

即\*

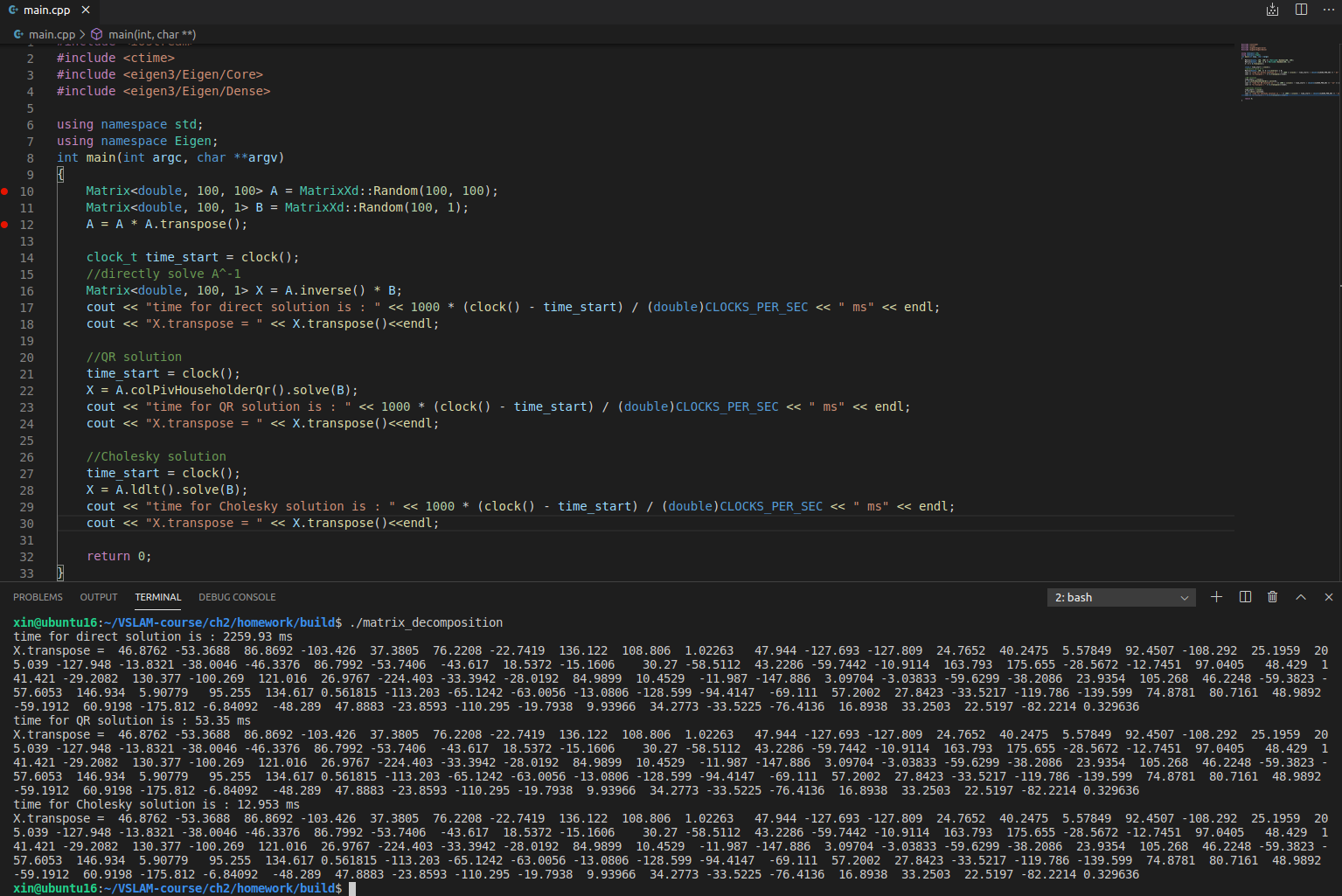
然后

另外易知对角元素 则可以得到(比如)

还有则可以得到(比如)

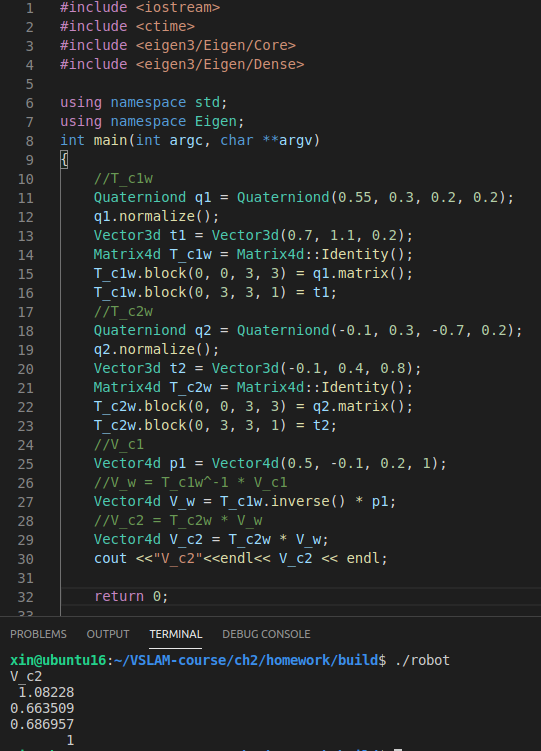
可以以此计算每个元素得到整个L矩阵

## 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。

直接法用时2259.93ms； QR分解53.35ms； Cholesky分解12.953ms

# 3 几何运算练习 (2 分,约 1 小时)

请编程实现此事,并提交你的程序。



# 4 旋转的表达 (2 分,约 1 小时)

## 设有旋转矩阵 R,证明且 det R = +1^2

假设有一组正交基构成坐标系1，经过一次旋转变成构成坐标系2,对于同一个向量a，他在两个坐标系下的坐标分别为和.

把这两个向量用基和系数的乘积来表示，则

两边同时左乘以,得到旋转矩阵定义如下

那我们反过来

两边同时乘以,得到旋转矩阵定义如下

可见。

同时证明旋转矩阵各列两两正交，举例来说,设。

以显然这两列的点积为0

则旋转矩阵是正交矩阵，其行列式等于。

## 设有四元数 q，我们把虚部记为 ε，实部记为 η，那么 q = (ε,η)。请说明 ε 和 η 的维度。

实部1维，虚部3维

## 请证明对任意单位四元数 q1, q2，四元数乘法可写成矩阵乘法：

正常向量情况的表达式

定义的

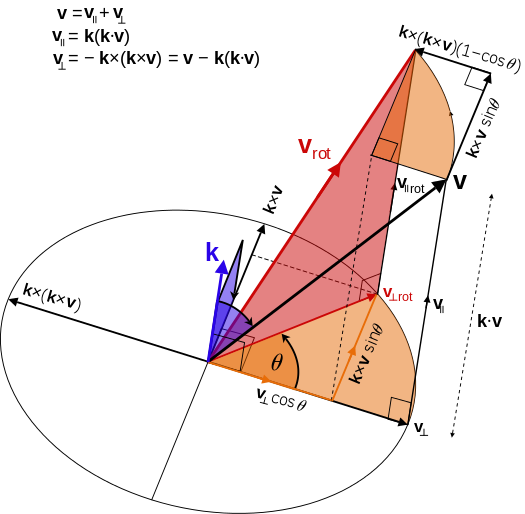
矩阵表达式结果跟上面向量表达式一样。

定义的

我们只需关注

即叉积的负交换律，则第一行部分和向量表达式第一行也一样。

# 5 罗德里格斯公式的证明 (2 分，约 1 小时)

[](https://en.wikipedia.org/wiki/File:Orthogonal_decomposition_unit_vector_rodrigues_rotation_formula.svg)

以上图为例，定义向量和绕k旋转之后得到向量，

推导如下:

# 6 四元数运算性质的验证 (1 分，约 1 小时)

## 此时 p′ 必定为虚四元数（实部为零）。请你验证上述说法

假设有点

代入四元数乘法向量形式得到

定义

剩下的

定义

现在只需要验证

我们忽略分母(另外，里俩-1抵消了)简化一下

## 上式亦可写成矩阵运算：p′ = Qp，请根据你的推导，给出矩阵 Q。

(注意，Q左上角是旋转矩阵R)

# 7 \* 熟悉 C++11 (2 分，约 1 小时)

**非静态数据成员初始化 int index = 0;**

C++11允许在定义的时候对非static非const成员初始化。

**拓展初始化列表 vector<A> avec{a1,a2,a3};**

以前只有数组可以使用初始化列表，现在其他容器可以用{}实现。

**lambda表达式 [](const A&a1, const A&a2){return a1.index<a2.index;}**

创建匿名的函数对象，简化工作

[]代表函数可以获得的全局变量,（）形参，->返回类型（此处没有定义）,{}函数体

**自动类型auto & a:avec**

auto 从初始化表达式中推断数据类型，比如auto i = 1.0;

编译时进行推导，不影响运行效率。

也不影响编译速度，因为本来也要判断右侧类型跟左侧是否匹配。

**range-base for loop for (auto & a:avec)**

有iterator的序列可以用这种简化的for循环