第二讲 三维空间的刚体运动

1、熟悉 Eigen 矩阵运算

设线性方程 $Ax=b$，在 A 为方阵的前提下，请回答以下问题：

1.1 在什么条件下，x 有解且唯一？

答案：非齐次线性方程组有唯一解的充要条件是 rank(A)=n。

1.2 高斯消元法的原理是什么？

答案：最基本的那种求方程解的方法，就是对矩阵进行行变换。

1.3 QR 分解的原理是什么？

答案：在了解QR分解之前，先了解一下Gram-Schmidt正交化：

存在可逆矩阵A的列向量组$(\alpha_1,\alpha_2,\alpha_3,...,\alpha_n)$经过正交化之后可以得到：

$$
\begin{aligned}
\varepsilon_1 &= \frac{\beta_1}{\parallel \beta_1 \parallel} = t_{11}\alpha_1 \\
\varepsilon_2 &= \frac{\beta_2}{\parallel \beta_2 \parallel} = t_{11}\alpha_1 + t_{22}\alpha_2 \\
&\vdots \\
\varepsilon_1 &= \frac{\beta_j}{\parallel \beta_j \parallel} = t_{1j}\alpha_1 + t_{2j}\alpha_2 + \cdots + t_{jj}\alpha_j
\end{aligned} \tag{1}
$$

把上述式子使用矩阵表示就可以得到下面的定义

> 对于物理意义，有时候不是那么重要，知道是这种数学表达就可以了。

定义：对于n阶方阵A，若存在正交矩阵Q和上三角矩阵R，使得$A=QR$，则该式称为矩阵A的完全QR分解或正交三角分解。(对于可逆矩阵A存在完全QR分解)。

$$
(\varepsilon_1, \varepsilon_2, \varepsilon_3, \ldots, \varepsilon_j) = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n) \begin{pmatrix} t_{11} & \ldots & t_{1n} \\ & \ddots & \vdots \\ 0 & & t_{nn} \end{pmatrix} \tag{2}
$$

QR基于我们熟悉的Gram-Schmidt正交化，令:

$$
\begin{gathered}
A = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n) \\
T = (t_{ij}) \\
Q = (\varepsilon_1, \varepsilon_2, \varepsilon_3, \ldots, \varepsilon_j)
\end{gathered} \tag{3}
$$

由此有$Q = AT$若记$R = T^{-1}$，则此时$A = QR$。其中Q是由Gram-Schmidt正交化得到的标准的正交基。

1.4 Cholesky 分解的原理是什么?

可以阅读Cholesky分解维基百科获取更加详细的解释，下面只是为了便于理解而简单的进行说明。

直接先说Cholesky 分解是用于求解$Ax = b$这个方程，然后具有等式:

$$
A = LL^T \tag{4}
$$

从式子可以看到就和代数的平方一样的效果，常用在优化里面作为误差；另外这个分解方法的优点是提高代数运算效率（矩阵求逆）、蒙特卡罗方法等场合中十分有用。

其中A是一个n阶厄米特正定矩阵(Hermitian positive-definite matrix)，L是下三角矩阵。**下面介绍推导过程**:

所以我们的目的是为了求L矩阵，算法由$i := 1$开始，令:

$$
A^{(1)} := A \tag{5}
$$

在步骤i中，矩阵$A^{(i)}$的形式如下:

$$
A^{(i)} = \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & \alpha_{i,j} & b_i^* \\ 0 & b_i & B^i \end{pmatrix} \tag{6}
$$

其中b_i^*表示b_i的共轭转置，若b是实数矩阵，b_i^*就是转置；I_{i-1}代表$i-1$维的单位矩阵。此时L_i定义为:

$$
L_i := \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & \sqrt{\alpha_{i,j}} & 0 \\ 0 & \frac{1}{\sqrt{a_{i,j}}} b_i & I_{n-1} \end{pmatrix} \tag{7}
$$

只有可以将矩阵$A^{(i)}$改写为:

$$
A^{(i)} = L_i A^{(i+1)} L_i^* \tag{8}
$$

我们发现这个和我们熟悉的的特征值分解结构相似$Q\Lambda Q$,接下来可以得到$A^{(i+1)}$的形式:

$$
A^{(i+1)} = \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & B^{(i)} - \frac{1}{a_{i,j}} b_i b_i^* \end{pmatrix} \tag{9}
$$

需要注意的是这里的$b_i b_i^*$是一个外积。

重复此步骤，直到i从1到n。n步之后，我们可以得到$A^{(n+1)} = I$。因此下三角矩阵L为:

$$
L := L_1 L_2 \ldots L_n \tag{10}
$$

1.5 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。**

```cpp
#include <iostream>

using namespace std;

#include <ctime>

// Eigen 部分
#include <Eigen/Core>
// 稠密矩阵的代数运算（逆，特征值等）
#include <Eigen/Dense>

#define MATRIX_SIZE 100


int main( int argc, char** argv )

{

    // 解方程
    // 我们求解 matrix_NN * x = v_Nd 这个方程
    // N的大小在前边的宏里定义，它由随机数生成
    // 直接求逆自然是最直接的，但是求逆运算量大

    cout <<"---解方程---"<<endl;
    clock_t time_stt = clock(); // 计时
    Eigen::Matrix< double, Eigen::Dynamic, Eigen::Dynamic > matrix_Dy;
    Eigen::Matrix< double, Eigen::Dynamic,  1> v_Nd;
    Eigen::Matrix< double, Eigen::Dynamic,  1> x;
    matrix_Dy = Eigen::MatrixXd::Random( MATRIX_SIZE, MATRIX_SIZE );
    matrix_Dy=matrix_Dy.transpose()*matrix_Dy;       //乔利斯基分解需要正定矩阵

    // 求逆

    v_Nd = Eigen::MatrixXd::Random( MATRIX_SIZE, 1 );
    x = matrix_Dy.inverse()*v_Nd;
    cout<<x[0]<<endl;
    cout <<"time use in normal inverse is " << 1000* (clock() - time_stt)/(double)CLOCKS_PER_SEC << "ms"<< endl;


    // Cholesky

    time_stt = clock();
    x = matrix_Dy.ldlt().solve(v_Nd);
    cout << x[0] << endl;
    cout <<"time use in Cholesky decomposition is " <<1000*  (clock() - time_stt)/(double)CLOCKS_PER_SEC <<"ms" << endl;

    // QR/Lu

    time_stt = clock();
    x = matrix_Dy.colPivHouseholderQr().solve(v_Nd);
    cout << x[0] << endl;
    x = matrix_Dy.fullPivLu().solve(v_Nd);
```

```
    cout << x[0] << endl;
    cout <<"time use in Qr decomposition is " <<1000*  (clock() - time_stt)/(double)CLOCKS_PER_SEC <<"ms" << endl;

    return 0;
}
```

2、几何运算练习

设有小萝卜1一号和小萝卜二号位于世界坐标系中。小萝卜一号的位姿为：$q1=[0.55,0.3,0.2,0.2]$,$t1=[0.7,1.1,0.2]^T$ （q 的第一项为实部）。这里的 q 和t表达的是 T_{cw}，也就是世界到相机的变换关系。小萝卜 二号的位姿为 $q_2=[-0.1,0.3,-0.7,0.2]$,$t_2=[-0.1,0.4,0.8]^T$ 。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。

```
#include <iostream>
#include <cmath>
using namespace std;

#include <Eigen/Core>
#include <Eigen/Geometry>

int main ( int argc, char** argv )

{
    Eigen::Isometry3d Tcw1= Eigen::Isometry3d::Identity();
    Eigen::Isometry3d Tcw2= Eigen::Isometry3d::Identity();

    Eigen::Quaterniond q1(0.55,0.3,0.2,0.2);//定义四元数Q1
    Eigen::Quaterniond q2(-0.1,0.3,-0.7,0.2);

    cout<<"quaternion q1 = \n"<<q1.coeffs() <<endl;  // 请注意coeffs的顺序是(x,y,z,w),w为实部，前三者为虚部

    cout<<"quaternion q2 = \n"<<q2.coeffs() <<endl;

    Eigen::Matrix<double, 3, 1> t1(0.7, 1.1, 0.2 );
    Eigen::Matrix<double, 3, 1> t2(-0.1, 0.4, 0.8 );

    Eigen::Matrix< double, 3, 1 > p1c(0.5, -0.1, 0.2);
    Eigen::Matrix< double, 3, 1 > p1w;
    Eigen::Matrix< double, 3, 1 > p2c;


    q1 = q1.normalized();
    q2 = q2.normalized();

    Tcw1.rotate ( q1.toRotationMatrix() );       // 按照rotation_vector进行旋转
    Tcw1.pretranslate (t1);                      // 平移向量
    cout << " Tcw1 Transform matrix = \n" << Tcw1.matrix() <<endl;

    Tcw2.rotate ( q2.toRotationMatrix() );      // 按照rotation_vector进行旋转
    Tcw2.pretranslate (t2);                      // 平移向量
    cout << "Tcw2 Transform matrix = \n" << Tcw2.matrix() <<endl;

```

```
    p1w = Tcw1.inverse()*p1c;
    p2c = Tcw2*p1w;
    cout << "p2c = \n" << p2c.matrix() <<endl;

    return 0;
}
```

3、旋转的表达

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。

3.1 设有旋转矩阵 R，证明 $R^TR = I$ 且 $detR = +1^2$

3.1.1 正交矩阵性质：

- 正交矩阵的每一个列向量都是单位向量，且向量之间两两正交。
- 正交矩阵的行列式为1或者-1.
- $A^-1 = A^T$ (充要条件)

3.1.2 证明旋转矩阵是正交矩阵且行列式为1

首先我们令空间的一个位置x_i经过旋转R和一次平移t之后得到新的位置x_i':

$$x_i\prime = Rx_i + t \tag{11}$$

对于每一个$i = 1, \ldots, n$有：

$$x_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{12}$$

需要注意的是旋转之后平移与平移之后旋转是不同的例如：

$$\begin{gathered} x\prime = R(x+s) = Rx + Rs = Rx + t \\ t = Rs \end{gathered} \tag{13}$$

由于旋转矩阵可以对任意形式的（列）向量组和进行变换，这里我们考虑沿着x，y和 z轴变换的情况，也就是笛卡尔坐标系上进行变换：

$$R\begin{pmatrix} x & y & z \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 \end{pmatrix} \tag{14}$$

因为一开始的（列）向量组$\begin{pmatrix} x & y & z \end{pmatrix}$是正交的，并且是单位向量；所以最后的结果$\begin{pmatrix} r_1 & r_2 & r_3 \end{pmatrix}$一定也是正交的单位向量组，然后一起组成了旋转向量$R$。

之后考虑R的转置R^T：

$$R^TR = \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} \begin{pmatrix} r_1 r_2 r_3 \end{pmatrix} \tag{15}$$

因为r_1, r_2和r_3都是单位向量并且相互正交，也就是$r_1 \cdot r_1 = 1, r_1 \cdot r_2 = 0$,因此：

$$R^TR = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{16}$$

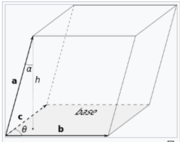
由此可以得到 $R^T = R^{-1}$，**即证旋转矩阵是正交的。**

对于行列式，我们知道矩阵的行列式是向量的混合积：$r_1 \cdot (r_2 \times r_3)$。这也表示以这些向量为边构成的平行六面体的体积。

以向量计算 [编辑]

另外一个方法是用向量 $\mathbf{a} = (a_1, a_2, a_3)$，$\mathbf{b} = (b_1, b_2, b_3)$，以及 $\mathbf{c} = (c_1, c_2, c_3)$ 来表示相交于一点的三条棱。平行六面体的体积 V 等于标量三重积：

$$V = |\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})| = |\mathbf{b} \cdot (\mathbf{c} \times \mathbf{a})| = |\mathbf{c} \cdot (\mathbf{a} \times \mathbf{b})|。$$



用向量来定义平行六面体。

证明：

以 $\mathbf{b}$ 和 $\mathbf{c}$ 来表示底面的边，则根据向量积的定义，底面的面积 A 为：

$$A = |\mathbf{b}||\mathbf{c}| \sin\theta = |\mathbf{b} \times \mathbf{c}|，$$

其中 θ 是 $\mathbf{b}$ 与 $\mathbf{c}$ 之间的角，而高为：

$$h = |\mathbf{a}| \cos\alpha，$$

其中 α 是 $\mathbf{a}$ 与 h 之间的角。

从图中我们可以看到，α 的大小限定为 $0^\circ \le \alpha < 90^\circ$。而向量 $\mathbf{b} \times \mathbf{c}$ 与 $\mathbf{a}$ 之间的角 β 则有可能大于90°（$0^\circ \le \beta < 180^\circ$）。也就是说，由于 $\mathbf{b} \times \mathbf{c}$ 与 h 平行，β 的值要么等于 α，要么等于 $180^\circ - \alpha$。因此：

$$\cos\alpha = \pm\cos\beta = |\cos\beta|，$$

且

$$h = |\mathbf{a}||\cos\beta|。$$

我们得出结论：

$$V = Ah = |\mathbf{a}||\mathbf{b} \times \mathbf{c}||\cos\beta|，$$

于是，根据标量积的定义，它等于 $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$ 的绝对值，即：

$$V = |\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})|。$$

证毕。

最后一个表达式也可以写成以下行列式的绝对值：

$$V = \left|\det \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}\right|。$$

由前面的结果可以得到R是正交矩阵，因此行列式为 +1或者-1；

emmm 看了很多解释，我觉得都不是很清除[Rotations and rotation matrices][2]，所以还是来算一下吧，具体看下面三种形式的旋转矩阵：

$$\mathrm{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}$$

$$\mathrm{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$\mathrm{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

简单的计算一下，例如$R_x(a)$按照第一行展开：

$$det(R_x(a)) = 1 * \left((cos\alpha)^2 + (sin\alpha)^2\right) == 1 \tag{17}$$

3.2 设有四元数 q，我们把虚部记为ε，实部记为 η，那么 $q = (\varepsilon, \eta)$。请说明 ε和 η 的维度[gaoxiang-cnblogs][3]

$$q = [\varepsilon, \eta], \quad \varepsilon = q_o \in R, \quad \eta = (\, q_1, q_2, q_3 \,) \in R^3 \tag{18}$$

3.3 四元数运算总结：

这里可以简单的扩展一下四元数相关的知识：

一个四元数可以写成如下形式：

$$q = \begin{pmatrix} a + id & -b - ic \\ b - ic & a - id \end{pmatrix} = aI + \; bi + \; cj + \; dk \tag{19}$$

其中：

$$I=\begin{pmatrix}1&0\\0&1\end{pmatrix},\ i=\begin{pmatrix}0&-1\\1&0\end{pmatrix},\ j=\begin{pmatrix}0&-i\\-i&0\end{pmatrix},\ k=\begin{pmatrix}i&0\\0&i\end{pmatrix} \tag{20}$$

这样子我们就可以得到四元数的一些性质：

$$\begin{cases} j^2=k^2=-1 \\ ij=k, jk=i, ki=j \\ ji=-k, kj=-i, ik=-j \end{cases} \tag{21}$$

对于四元数的运算（加法、乘法、共轭、模）有：

$$\begin{cases} q=[\varepsilon,\eta],\ \ \varepsilon=q_o\in R,\ \ \eta=(q_1,q_2,q_3)\in R^3 \\ p+q=[\varepsilon_p+\varepsilon_q,\eta_p+\eta_q] \\ pq=[\varepsilon_p,\eta_p][\varepsilon_q,\eta_q]=[\varepsilon_p\varepsilon_q-\eta_p\cdot\eta_q,\ \varepsilon_p\eta_q+\varepsilon_q\eta_p+\eta_p\times\eta_q] \\ p\cdot q=p_0q_0+p_1q_1i+p_2q_2j+p_3q_3k \\ \overline{p}=[\varepsilon_p,-\eta_p] \\ \mid q\mid=\mid[s,v]\ \mid=\sqrt{s^2+\mid v\mid^2} \end{cases} \tag{22}$$

所以对于 pq是指四元数每个位置上的数值分别相乘经过一顿操作可以写成上面的式子的简洁形式**注意与点乘区分。**

其中：

$$\eta_p\times\eta_q=\begin{vmatrix} i&j&k\\ p_1&p_2&p_3\\ q_1&q_2&q_3\end{vmatrix} \tag{23}$$

$$\eta_p\cdot\eta_q=p_1q_1+p_2q_2+p_3q_3 \tag{24}$$

4、罗德里格斯公式的证明

[参考： https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula](https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula)

罗德里格斯公式提供了一种算法，可以计算从$so(3)$（$SO(3)$的李代数）到$SO(3)$的指数映射，而无需实际计算完整矩阵指数。令$v\in R^3$然后k是一个单位向量，描述了根据右手定则围绕其旋转角度θ的旋转轴，例如$[1\ 0\ 0]^T\ [0\ 1\ 0]^T\ [0\ 0\ 1]^T$，那么旋转向量$v_{rot}$的罗德里格斯公式的形式如下：

$$v_{rot}=vcos\theta+(k\times v)sin\theta+k(k\cdot v)(1-cos\theta) \tag{25}$$

首先，使用点积和叉积，向量v可以分为与轴k平行和垂直的分量：

$v=v_{\parallel}+v_{\perp}$

其中与k平行的分量为：$v_{\parallel}=(v\cdot k)k$

垂直于k的分量是v在k上的向量投影：$v_{\perp}=v-v_{\parallel}=v-(k\cdot v)k=-k\times(k\times v)$

向量$k\times v$可以看作是$v_{\perp}$的副本，逆时针绕k旋转了90°，因此它们的大小相等，但方向垂直。

同理可得，向量$k\times$（$k\times v$）是将v的副本绕k逆时针旋转到180°，因此$k\times$（$k\times v$）和$v_{\perp}$的大小相等，但方向相反，所以是负号。

展开向量三乘积（vector triple product）可建立平行分量和垂直分量之间的连接，给定任何三个向量a，b，c公式的公式为$a \times (b \times c) = (a \cdot c)b - (a \cdot b)c$。

平行于轴的分量在旋转下不会改变大小或方向:$v_{\parallel rot} = v_{\parallel}$

只有垂直分量会改变方向，但保持其大小:

$$\begin{cases} |v_{\perp rot}| = |v_{\perp}|, \\ v_{\perp rot} = cos\theta v_{\perp} + sin\theta k \times v_{\perp} \end{cases} \tag{26}$$

由于 k and $v_{\parallel}$ 是平行的，因此他们的叉积为0：

$$\begin{cases} k \times v_{\perp} = k \times (v - v_{\parallel}) = k \times v - k \times v_{\parallel} = k \times v \\ v_{\perp rot} = cos\theta v_{\perp} + sin\theta k \times v \end{cases} \tag{27}$$

旋转分量的形式类似于笛卡尔基础上二维平面极坐标（r，θ）中的径向矢量:

$$r = rcos\theta e_x + rsin\theta e_y \tag{28}$$

其中ex，ey是其指示方向上的单位向量。

现在完整的旋转向量为：

$$v_{rot} = v_{\parallel rot} + v_{\perp rot} \tag{29}$$

通过将$v_{\parallel rot}$和$v_{\perp rot}$的定义代入等式，得出:

$$\begin{aligned} v_rot &= v_{\parallel} + cos\theta v_{\perp} + sin\theta\,(k \times v) \\ &= v_{\parallel} + cos\theta(v - v_{\parallel}) + sin\theta\,(k \times v) \\ &= cos\theta v + (1 - cos\theta)v_{\parallel} + sin\theta\,(k \times v) \\ &= cos\theta v + (1 - cos\theta)(k \cdot v)k + sin\theta\,(k \times v) \end{aligned} \tag{30}$$

参考链接：

- https://www.cnblogs.com/indulge-code/p/10492209.html "东电逸仙-cnblog"
- https://onlinelibrary.wiley.com/doi/pdf/10.1107/S0907444901012410 "Rotations and rotation matrices"
- https://www.cnblogs.com/gaoxiang12/p/5120175.html " gaoxiang-cnblogs"