2023-2024 学年第二学期

武汉大学测绘学院导航工程专业

导航学课程设计实验报告

课程名称： 导航学

专业班级： 智能导航一班

姓　　名： 秦旗峰

学　　号： 2023302143029

实验名称： 编写导航常用坐标系统坐标转换程序

指导教师： 辜声峰 **老师**

实验时间： 2024.5

目录

1. 实验目的

大地坐标系、空间直角坐标系、站心坐标系 NED(l或n系)和载体坐标系 FRD(b)系都是导航过程中常用的坐标系。不同的坐标系有着不一样的作用和使用场景，本次的实验任务旨在加强我们对不同坐标系的理解，和强化利用编程解决不同坐标系之间转换问题的能力。本次实验需要利用编程完成由空间直角坐标系向站心坐标系的转换以及完成由站心坐标系向载体坐标系的转换。

2. 数学原理

2.1 常用坐标系

在导航与定位研究中，坐标系用来描述载体的运动，因此导航技术均需要在坐标系的基础上实现。

下面介绍在本实验中使用到的坐标系：

空间直角坐标系也称**地心地固坐标系**（Earth Center Earth Fixed，ECEF），简称e系，坐标表示为$(X，Y，Z)$这是现代大地测量与导航中常用的一种坐标系。该坐标系的坐标原点位于地球椭球的中心;z轴与地球椭球短轴(旋转轴)一致，指向北极(N);以起始大地子午面与椭球赤道面的交线为x轴，指向起始子午线;y轴垂直于x轴和y轴组成右手坐标系(如图 2.1)是一种随地球自转而转动的地球固连坐标系。空间某点的位置用三维直角坐标（$X，Y，Z$）来表示。

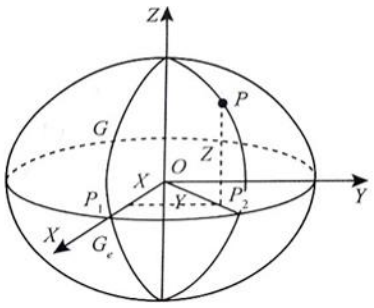

图 2.1 地心地固坐标系

站心坐标系，也常被称作**导航坐标系（Navigation）**，简称n系，也就是 NED，一般以运动载体中心为原点，X 轴指向地理北方向，Y 轴指向东方向，Z 轴垂直于大地椭球面向下（如图 2.2），构成右手坐标系，记为“北东地”坐标系，常用的还有“东北天”“北西天”等右手系。

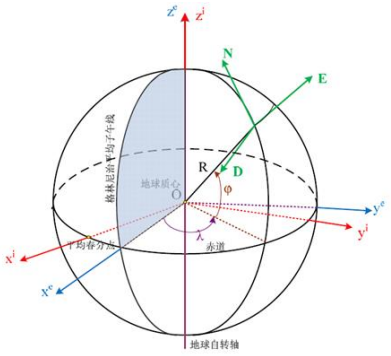

图 2.2　绿色为站心坐标系

载体坐标系（Body），简称b系，载体坐标系的原点与载体质心重合，一般以 IMU 中心为原点，X轴指向载体运动前方，Z 轴垂直地面向下或向上，Y轴指向载体运动前进方向的两侧，与X轴和Y构成右手系，一般简称前-右-下坐标系和右-前-上坐标系（如图 2.3）。

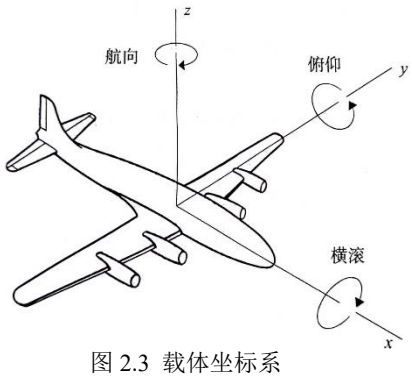

图 2.3 载体坐标系

2.1 姿态参数化

姿态信息反应载体坐标系相对于参考坐标系的角度旋转信息，对姿态的参数化方法较多，常用的包括**欧拉角（Euler Angle）**、**四元数（Quaternion）**、**方向余弦矩阵（Direction Cosine Matrix，DCM）**和**等效旋转矢量（Rotation Vector）**等。

欧拉角是一种直观的描述两个坐标系之间相对旋转的方法，它把一个旋转分解为绕 3 个不同轴的旋转。由于绕不同轴旋转有多种排列顺序，因此欧拉角有着多种不同的定义方法。常用**横滚角**（Roll）、**俯仰角**（Pitch）和**航向角**（Yaw）来描述载体坐标系相对于导航坐标系的姿态。欧拉角的一个重大缺陷是当俯仰角为±90°时，无法区分横滚角和航向角，即欧拉角姿态表示法存在奇异性问题。

方向余弦矩阵常用于三维空间坐标旋转，

绕Z轴旋转α角：

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

绕Y轴旋转β角：

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

绕X轴旋转γ角：

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

注意旋转矩阵乘积方式：根据旋转次序依次左乘。例如下式表示坐标系 2 是由坐标系 1 依次绕X、Y、Z轴旋转得到：$r_2=R_z(\alpha)R_y(\beta)R_x(\gamma)r_1$。每个子旋转矩阵都是正交矩阵，即$R=R^T$。

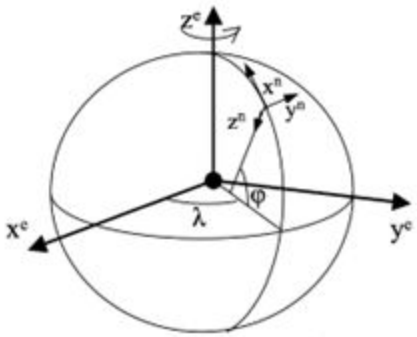

图 2.4 地心地固坐标系到站心坐标系

• 在从地心地固坐标系坐标系转换到站心坐标系（如图 2.4）过程中，空间直角坐标系到北东地地理标系（站心坐标系）可通过绕 Ze 轴转动λ ，再绕所得的坐标系 Ye 轴转动（270°-φ），其中λ和φ分别为大地经度和纬度因此从空间直角坐标系坐标X_e变换到站心坐标系坐标X_l的旋转矩阵：

$$C_e^l=R_y(270°-\varphi)R_z(\lambda)=\begin{bmatrix} -sin\varphi & 0 & cos\varphi \\ 0 & 1 & 0 \\ -cos\varphi & 0 & -sin\varphi \end{bmatrix}\begin{bmatrix} cos\lambda & sin\lambda & 0 \\ -sin\lambda & cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

即：

$$C_e^l=\begin{bmatrix} -sin\varphi cos\lambda & -sin\varphi sin\lambda & cos\varphi \\ -sin\lambda & cos\lambda & 0 \\ -cos\varphi cos\lambda & -cos\varphi sin\lambda & -sin\varphi \end{bmatrix}$$

因此，已知测站和目标点的空间直角坐标系坐标分别为X_{origin}^e和X_{target}^e则，目标点在以测站为中心的站心坐标系下的坐标可以表示为：

$$X_{target}^l=C_e^l(X_{target}^e-X_{origin}^e)$$

• 从站心坐标系到载体坐标系需要确定载体的欧拉角（姿态角）$[\phi \quad \theta \quad \psi]^T$，分别对应**横滚角(Roll)**、**俯仰角(Pitch)**和**航向角(Yaw)**。站心坐标系与载体坐标系原点相同时，从站心坐标

系变换到载体坐标系只与姿态角相关（如图 2.5），以惯导元件（IMU）为坐标原点，建立前-右-下载体坐标系，其姿态角为$[\phi \quad \theta \quad \psi]^T$。

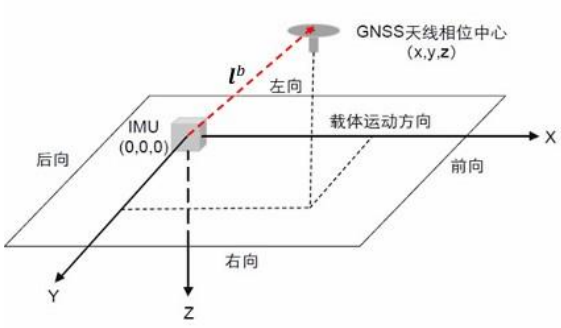

图 2.5 以 IMU 为原点的载体坐标系

从站心坐标系变换到载体坐标系需要先绕方位轴z轴转动ψ，再将得到的坐标轴绕 y 轴转动θ，最后绕着 x 轴转动ϕ.变换矩阵可以表示成：

$$C_l^b = R_x(\phi)R_y(\theta)R_z(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix}\begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

即：

$$C_l^b = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ sin\phi sin\theta cos\psi - cos\phi sin\psi & sin\phi sin\theta sin\psi + cos\phi cos\psi & sin\phi cos\theta \\ cos\phi sin\theta cos\psi + sin\phi sin\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi & cos\phi cos\theta \end{bmatrix}$$

若已知惯导 IMU 和 GNSS 天线相位中心的空间直角坐标系坐标P_{INS}^e和P_{GNSS}^e，则在空间直角坐标系下向量表示为$P_{GNSS}^e - P_{INS}^e$，依据变换公式，GNSS 相位中心在 IMU 载体坐标系下坐标$l_{bg}^b = C_e^b(P_{GNSS}^e - P_{INS}^e)$,其中$C_e^b$可以由上式两个变换矩阵相乘得到：$C_e^b = C_l^b C_e^l$，即可求得相应坐标。

3. 程序设计

作业要求：

- 题目：已知站心X_o空间直角坐标(-2148744.679 4426641.[sn] 4044655.862)，与目标点X_t空间直角坐标(-2148745.679 4426639.122 4045655.[pn])，其中[sn]与[pn]分别用学号和手机号的末尾三位进行代替，试求出目标点在站心坐标系下的坐标
- 题目：已知IMU(惯性测量单元)的欧拉角为Roll=30deg，Pitch=45deg，Yaw=120deg。IMU中心的空间直角坐标(-2148746.822 4426645.265 4044653.164)，GNSS天线相位中心的空间直角坐标(-2148746.708 4426645.350 4044653.894)，根据以上条件，试求出在以IMU为中心的载体坐标系下，GNSS天线相位中心的坐标

3.1 函数命名

3.1.1 矩阵函数

```
class Matrix
{
public:
    void ttmatrix(double(*a)[3], double(*b)[3], double(*c)[3]);    //声明三阶矩阵乘三阶矩阵计算函数
    void tomatrix(double(*d)[3], double *e, double*f);    //声明三阶矩阵成三维列向量计算函数
};
```

在此命名了两个矩阵函数，ttmatrix（double(*a)[3], double(*b)[3], double(*c)[3])是两个三阶矩阵相乘的函数，表示$a_{3\times3}b_{3\times3}=c_{3\times3}$，tomatrix（double(*d)[3], double *e, double*f)是三阶矩阵与三维列向量相乘，表示$d_{3\times3}e_{3\times1}=f_{3\times1}$。

3.1.2 坐标函数

```
class Coordinate
{
public:
    void BLH2XYZ(double*m, double*n);    //声明从BLH转换到XYZ函数
    void XYZ2BLH(double*c, double*d);    //声明从XYZ转换到BLH函数
    void XYZ2NED(double*p, double*q,double*w);    //声明从XYZ转换到NED（站心坐标系）的函数
    void NED2FRD(double*x, double *t, double*a, double*g);    //声明从NED转换到FRD（载体坐标系）的函数
};
```

对于坐标变换，新命名了两个函数：空间直角坐标系到站心坐标系函数XYZ2NED(double*p , double*q , double*w)，p 和 q 分别为站心(origin)和目标点(target)的空间直角坐标系坐标，w 为 target 在站心坐标系下的坐标，站心坐标系到载体坐标系函数NED2FRD(double*x , double*t , double*a , double*g)，x 和 t 分别为 IMU 和 GNSS 相位中心的空间直角系坐标，a 为 IMU 的偏转角，g 表示 GNSS 相位中心在 IMU 载体坐标系下的坐标。

3.2 输入与输出参数

3.2.1 矩阵函数

①ttmatrix 函数

```
void Matrix::ttmatrix(double(*a)[3], double(*b)[3], double(*c)[3])    //定义三阶矩阵与三阶矩阵的乘法
{
    int x = 0, y = 0;
    for (double *p = c[0]; p < c[0] + 9; p++)
    {
        *p = 0;
    }
    for (int i = 0; i < 9; i++)    //循环9次，求9个数
    {
        for (int j = 0; j < 3; j++)    //一个小循环完成[x][y]元素的求值
        {
            *(c[x] + y) += *(a[x] + j)*(*(b[j] + y));    //累加求每一项
        }
        y++;    //换到下一项
        if (i == 2 || i == 5 || i == 8)x++;    //换行
        if (y == 3)y = 0;    //一行求完后从下一行第一个开始求
    }
}
```

输入 a[3][3]和 b[3][3]，得到 c[3][3]。

②tomatrix 函数

```
void Matrix::tomatrix(double(*d)[3], double *e, double*f)      //定义三阶矩阵与三维列向量的乘法
{
    for (int i = 0; i < 3; i++)
    {
        *(f + i) = 0;
    }
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            *(f + i) += *(d[i] + j)*(*(e + j));
        }
    }
}
```

输入 d[3][3]和 e[3]，得到 f[3]。

3.2.2 坐标函数

①XYZ2NED 函数

```
void Coordinate::XYZ2NED(double*Xo, double*Xt,double*Xl)    //定义XYZ到NED坐标转换函数，Xo，Xt分别是站心和目标点的空间直角坐标，Xl表示目标点的站心坐标
{
    double BO[3], m[3];    //BO是Xo站心的大地坐标系，m是一个中间量
    XYZ2BLH(Xo, BO);       //将站心的空间直角坐标转换成大地坐标
    double B = BO[0] * pi / 180, L = BO[1] * pi / 180;
    double Cel[3][3] = { {-sin(B)*cos(L),-sin(B)*sin(L),cos(B)},{-sin(L),cos(L),0},{-cos(B)*cos(L),-cos(B)*sin(L),-sin(B)} };    //坐标变换矩阵
    for (int i = 0; i < 3; i++)
    {
        m[i] = *(Xt+i) - *(Xo+i);    //目标点和站心的坐标向量
    }
    Run.tomatrix(Cel, m, Xl);          //Xl=Cel*m,进行坐标矩阵的变换
}
```

输入 Xo[3]（站心的空间直角系坐标），和 Xt[3]（目标点的空间直角坐标系坐标），得到 Xl[3]（目标点在站心坐标系下的坐标）.

②NED2FRD 函数

```
void Coordinate::NED2FRD(double*x, double *t, double*A, double*g)    //定义NED到FRD坐标转换函数，x和t分别为载体和目标点的空间直角坐标，A表示载体的欧拉角，g表示目标点的载体坐标
{
    double R = (*A)*pi / 180, P = *(A + 1)*pi / 180, Y = *(A + 2)*pi / 180;
    double Bx[3],deta[3],Ceb[3][3];        //Bx表示载体的大地坐标，deta是中间量，接收目标点和载体的坐标向量，Ceb是坐标变换矩阵相乘的结果
    XYZ2BLH(x, Bx);              //将载体的空间直角坐标转换成大地坐标
    double B = Bx[0] * pi / 180, L = Bx[1] * pi / 180;
    double Cel[3][3] = { {-sin(B)*cos(L),-sin(B)*sin(L),cos(B)},{-sin(L),cos(L),0},{-cos(B)*cos(L),-cos(B)*sin(L),-sin(B)} };    //坐标变换矩阵
    double Clb[3][3] = { {cos(P)*cos(Y),cos(P)*sin(Y),-sin(P)},{sin(R)*sin(P)*cos(Y) - cos(R)*sin(Y),sin(R)*sin(P)*sin(Y) + cos(R)*cos(Y),sin(R)*cos(P)},{cos(R)*sin(P)*cos(Y) + sin(R)*sin(Y),cos(R)*sin(P)*sin(Y) - sin(R)*cos(Y),cos(R)*cos(P)} };
    for (int i = 0; i < 3; i++)
    {
        deta[i] = *(t + i) - *(x + i);    //目标点和载体的坐标向量
    }
    Run.ttmatrix(Clb, Cel, Ceb);    //Ceb=Clb*Cel
    Run.tomatrix(Ceb, deta, g);     //g=Ceb*deta,进行坐标变换
}
```

输入 x[3]（IMU 的空间直角坐标系坐标）、t[3]（GNSS 相位中心的空间直角坐标系坐标）和 A[3]（IMU 的偏转角）,得到 g[3]（GNSS 相位中心在 IMU 载体坐标系下的坐标）。

3.2.3 main 函数

```
int main()
{
    Matrix Run;
    Coordinate Transfer;    //声明坐标转换对象Transfer
    double Xo[3] = { -2148744.679,4426641.029,4044655.862 }, Xt[3] = { -2148745.679,4426639.122,4045655.657 };        //站心和目标点的空间直角坐标系
    double Xl[3];          //Xl[3]用于接收坐标变换后的结果，表示目标点在站心坐标系下的坐标
    Transfer.XYZ2NED(Xo, Xt, Xl);        //调用类函数XYZ2NED，实现从空间直角坐标系到站心坐标系的转换
    cout << "目标点在站心坐标系下坐标为:(";
    for (int i = 0; i < 3; i++)
    {
        cout << setiosflags(ios::fixed)<<setprecision(10)<< Xl[i];        //输出坐标
        if (i != 2)cout << ",";
    }
    cout << ")" << endl;
    double XI[3] = { -2148746.822,4426645.265,4044653.164 }, XG[3] = { -2148746.708,4426645.350,4044653.894 };        //惯导空间直角坐标系XI，GNSS接收机的空间直角坐标系
    double Ola[3] = { 30,45,120 }, lbg[3];        //欧拉角Ola，lbg用于接收坐标转化后的结果，表示GNSS接收机在惯导载体坐标系下的坐标
    Transfer.NED2FRD(XI, XG, Ola, lbg);          //进行坐标转换
    cout << "GNSS相位中心在惯导载体坐标系下坐标为:(";
    for (int i = 0; i < 3; i++)
    {
        cout << setiosflags(ios::fixed)<<setprecision(8) << lbg[i] ;        //输出结果
        if (i != 2)cout << ",";
    }
    cout << ")" << endl;
    return 0;
}
```

在 main 函数中定义了各个点的空间直角坐标系坐标，再根据题目要求调用已经定义的函数。

3.3 返回值

除了主函数是 int 类型返回 0 值之外，其余函数为 void 类型，无返回值。

3.4 单位

输入单位均为米（m），和度（°），输出结果均为米（m）

4. 实验结果与分析

```
目标点在站心坐标系下坐标为：(771.0749528788,1.7323712370,-636.4236753976)
GNSS相位中心在惯导载体坐标系下坐标为：(0.06526932,-0.65955607,-0.33741177)
```

转换结果

1.732371237 , 771.0749528788 , 636.4236753976

单位：米（m）

题目一	N	E	D
编程结果	771.0749528788	1.7323712370	-636.4236753976
网站结果	771.0749528788	1.732371237	-636.4236753976

与网站解算答案（在 WGS84 参考大地椭球下运算结果）对比，程序解算已经有较高精度。

题目二	x	y	z
编程结果	0.06526932	-0.65955607	-0.33741177

5. 总结

本实验涉及到的导航坐标系均为生活中常用的坐标系，不同的坐标系应用范围不同；应用场景不同，因此实现坐标系之间的相互转换尤为重要。本次实验对大地坐标系、地心地固坐标系、载体坐标系和站心坐标系的相互转换进行了深入了解，加深了载体坐标系和站心坐标系的概念以及定义方式，熟悉了坐标系间的旋转变换矩阵，在提高专业素养的同时，提高了数学、编程能力。