一、群的性质

1. {Z, +}是否为群？若是,验证其满足群定义;若不是,说明理由。其中 Z 为整数集。

回答:

是

根据群的四性质：

1. 封闭性：任意两个整数做加法运算，仍然是属于整数集合
2. 结合律：任意三个整数做加法运算，他们相加的先后顺序不影响最终结果
3. 幺元：0与任意整数做加法，该整数的值不变
4. 逆：任意整数的逆为其相反数（在加法运算的集合内），那么-Z + Z = 0 对任意整数都成立，0也是整数。

2. {N, +}是否为群？若是,验证其满足群定义;若不是,说明理由。其中 N 为自然数集。

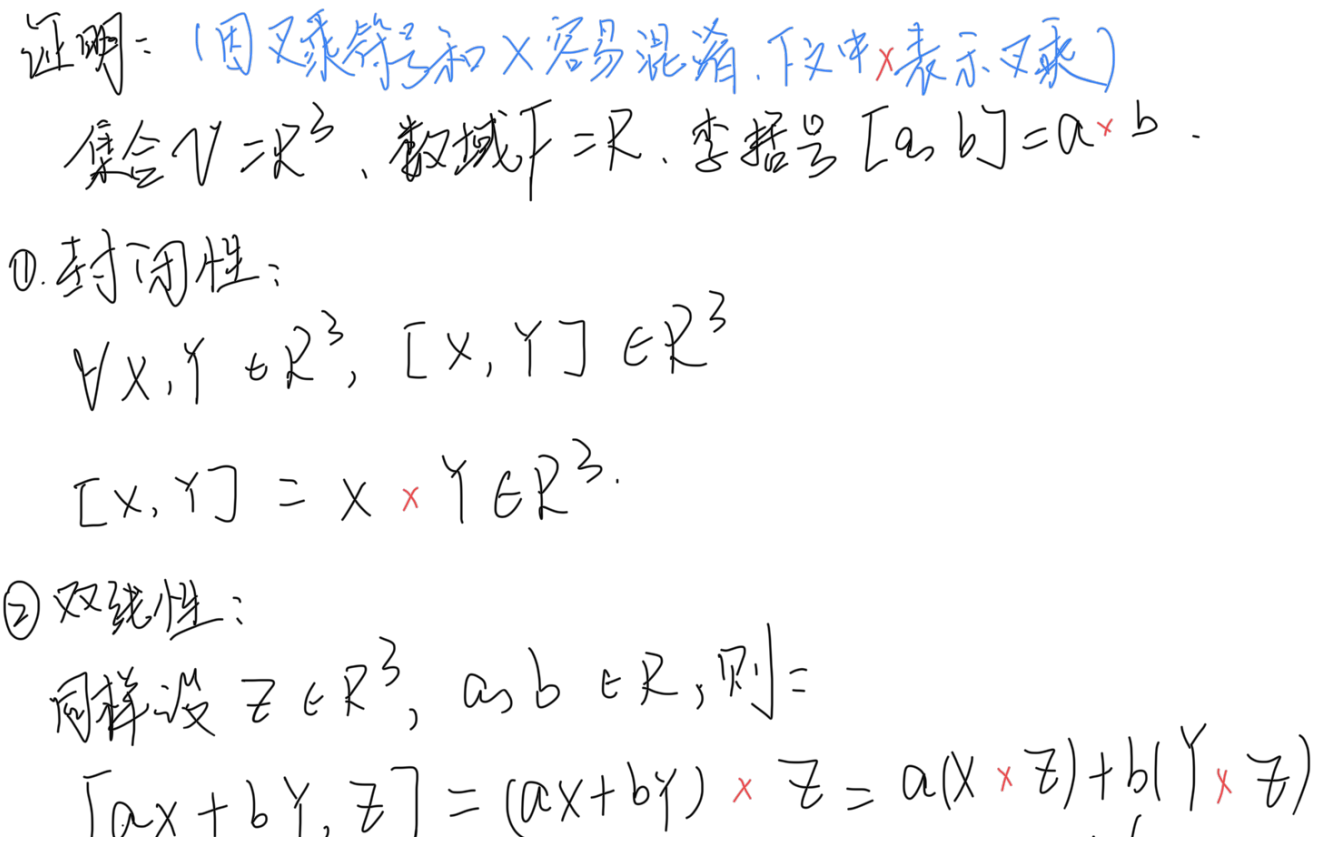
回答:

不是，因为不满足第四条逆的规则。

自然数是非负整数，正整数的逆为负整数不属于自然数集合内。

二、验证向量叉乘的李代数性质

回答:



$[z, ax+by] = z\times(ax+by) = a(z\times x)+b(z\times y)$

$= a[z,x] + b[z,y]$

③自反性：

$[x, x] = x \times x = 0$

④雅可比等价：

$[x,[y,z]]+[y,[x,z]]+[z,[x,y]]$

$= x\times y\times z + y\times x\times z + z\times x\times y$

$= 0.$ （这里利用三向量相乘公式）

$\vec{a}\times\vec{b}\times\vec{c} = \vec{a}\cdot(\vec{c}\cdot\vec{b}) - \vec{a}(\vec{b}\cdot\vec{c})$

故向量叉乘满足李代数性质.

三、推导SE(3)的指数映射

回答：

证明：

1. $\xi^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ 0^T & 0 \end{bmatrix}$ $\xi^\wedge\cdot\xi^\wedge = \phi^\wedge\cdot\xi^\wedge$

$\xi^\wedge\cdot\xi^\wedge\cdot\xi^\wedge = (\phi^\wedge)^2\xi^\wedge$

$\exp(\xi^\wedge) = \sum_{n=0}^{\infty}\frac{(\xi^\wedge)^n}{n!} = I + \xi^\wedge + \frac{(\xi^\wedge)^2}{2!} + \cdots$

$$= I + \xi^\wedge + \frac{\phi^\wedge \xi^\wedge}{2!} + \frac{(\phi^\wedge)^2 \xi^\wedge}{3!} + \cdots$$

$$= I + \left[\frac{(\phi^\wedge)^0}{1!} + \frac{(\phi^\wedge)^1}{2!} + \frac{(\phi^\wedge)^2}{3!} + \cdots\right] \xi^\wedge$$

$$= I + \left[\sum_{n=0}^{\infty} \frac{(\phi^\wedge)^n}{(n+1)!}\right] \cdot \xi^\wedge$$

$$= \begin{bmatrix} \sum_{n=0}^{\infty} \frac{(\phi^\wedge)^n}{n!} & \sum_{n=0}^{\infty} \frac{(\phi^\wedge)^n}{(n+1)!} \cdot \rho \\ 0^T & 1 \end{bmatrix}$$

第一部分证明完成，接下来主要是左扰动雅克比的推导.

2. $J = \sum_{n=0}^{\infty} \frac{(\phi^\wedge)^n}{(n+1)!}$

角度

$\phi = \theta \vec{a}$

单位向量

同推导SO3一致，我们令

其中 $\begin{cases} \vec{a}^\wedge \vec{a}^\wedge = \vec{a}\vec{a}^T - I \\ \vec{a}^\wedge \vec{a}^\wedge \vec{a}^\wedge = -\vec{a}^\wedge \end{cases}$

$J = \frac{(\theta \vec{a}^\wedge)^n}{\theta \cdot \vec{a}^\wedge}, \quad \frac{\theta^2 \cdot (\vec{a}^\wedge)^2}{\theta^3 \cdot a^\wedge}$

$$J = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} = 1 + \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!}$$

$$- \frac{\theta^4 \cdot (\vec{a}^\wedge)^2}{5!} + \frac{\theta^5 \cdot \vec{a}^\wedge}{6!} + \cdots$$

$$= 1 + \left[-\frac{1}{\theta} \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \frac{\theta^8}{8!} + \cdots \right) + \frac{1}{\theta} \right] \vec{a}^\wedge$$

$$+ \left[-\frac{1}{\theta} \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \cdots \right) + 1 \right] \cdot (\vec{a}^\wedge)^2$$

$$= 1 + \frac{1 - \cos\theta}{\theta} \vec{a}^\wedge + \frac{\theta - \sin\theta}{\theta} \cdot (\vec{a}^\wedge)^2$$

$$= 1 + \frac{1 - \cos\theta}{\theta} \vec{a}^\wedge + \frac{\theta - \sin\theta}{\theta} \left(\vec{a} \vec{a}^T - I \right)$$

$$= \frac{\sin\theta}{\theta} I + \left(1 - \frac{\sin\theta}{\theta} \right) \vec{a} \vec{a}^T + \frac{1 - \cos\theta}{\theta} a^\wedge$$

四、证明SO(3)的伴随性质

回答:

1. 首先：

$$R\xi^{\wedge}R^{T}\vec{v}=R\cdot(\xi\times(R^{T}\cdot\vec{v}))=(R\cdot\xi)\times(RR^{T}\vec{v})$$

$$=(R\cdot\xi)\times\vec{v}=(R\xi)^{\wedge}\cdot\vec{v}$$

故可知：$R\xi^{\wedge}R^{T}=(R\xi)^{\wedge}$

2、$R\exp(\xi^{\wedge})R^{T}=R\sum_{n=0}^{\infty}\frac{(\xi^{\wedge})^{n}}{n!}R^{T}$

$$=\sum_{n=0}^{\infty}\frac{R\cdot(\xi^{\wedge})^{n}R^{T}}{n!}=\sum_{n=0}^{\infty}\frac{R\xi^{\wedge}R^{T}\cdot R\xi^{\wedge}R^{T}\cdots R\xi^{\wedge}R^{T}}{n!}$$

n'个ξ

$$=\sum_{n=0}^{\infty}\frac{(R\cdot\xi^{\wedge}R^{T})^{n}}{n!}$$

根据1中推导知$R\xi^{\wedge}\cdot R^{T}=(R\cdot\xi)^{\wedge}$

故上式$=\sum_{n=0}^{\infty}\frac{(R\xi)^{\wedge})^{n}}{n!}=\exp[(R\xi)^{\wedge}]$

证毕.

五、轨迹的描绘

1. Twc 的平移部分构成机器人的轨迹，它的物理意义是什么？为什么画出Twc的平移部分就得到机器人的轨

迹?

回答:

Twc的平移部分就是机器人相对于世界坐标系的坐标点，所以画出Twc之后，就等于在世界坐标系下，画出机器人一系列的坐标点，连接起来就是机器人的轨迹图。我们在观察机器人移动的轨迹其实就是在看机器人相对map坐标系的位置。

2. 使用提供好的轨迹文件和画图程序，完成数据读取部分代码，书写CMakeLists.txt让程序运行起来。

CMakeLists.txt文件

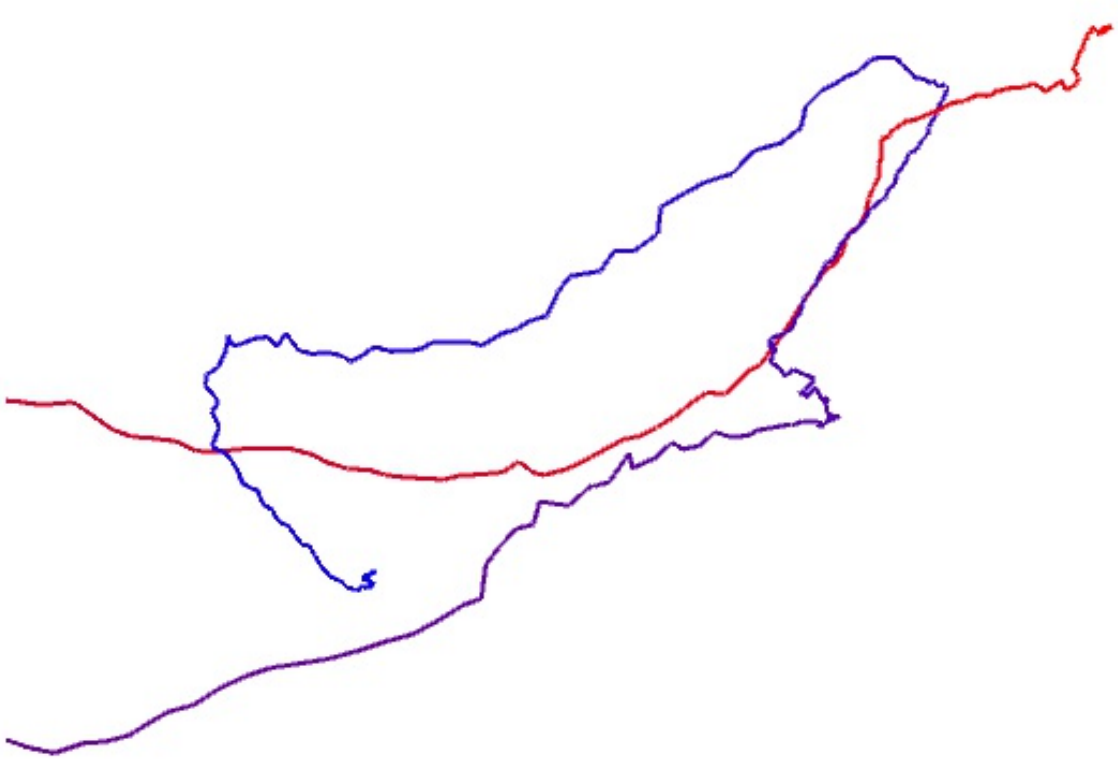
```
cmake_minimum_required(VERSION 2.8)
project(homework3)

# 下面这句保证pangolin正常编译通过
set(CMAKE_CXX_FLAGS "-std=c++11")

find_package(Pangolin REQUIRED)
find_package( Sophus REQUIRED )

include_directories( ${Sophus_INCLUDE_DIRS} )
include_directories(
        ${PROJECT_SOURCE_DIR}
        ${Pangolin_INCLUDE_DIRS}
)

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/MyWork)
add_executable(draw_trajectory draw_trajectory.cpp)

target_link_libraries(draw_trajectory ${Pangolin_LIBRARIES} ${Sophus_LIBRARIES})
```

主函数部分cpp文件

```
// path to trajectory file
string trajectory_file = "../trajectory.txt";


// function for plotting trajectory, don't edit this code
// start point is red and end point is blue
void DrawTrajectory(vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>>);

int main(int argc, char **argv) {

    vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>> poses;

    /// implement pose reading code
    // start your code here (5~10 lines)
    Eigen::Quaterniond q;
    Eigen::Vector3d t;
    Sophus::SE3 T;
    ifstream trajectory;
    double time_stamp;

    trajectory.open(trajectory_file.c_str());
    if (!trajectory.is_open()){
        cout << "the file is empty!!" << endl;
        return -1;
    }
```

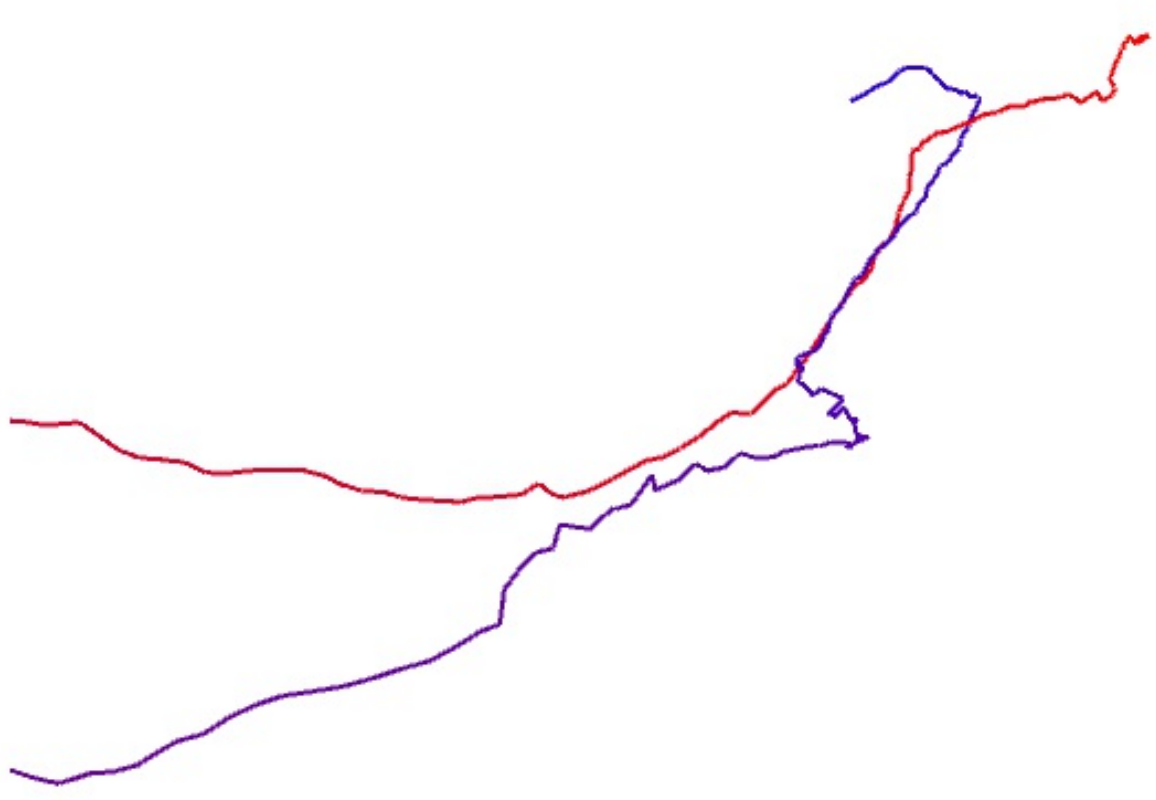
```
    string sLine;
    while(getline(trajectory, sLine) && !sLine.empty()){
        istringstream iss(sLine);
        iss >> time_stamp >> t[0] >> t[1] >> t[2] >> q.x() >> q.y() >> q.z() >>
q.w();
        T = Sophus::SE3(q, t);
        poses.push_back(T);
    }
    trajectory.close();

    // end your code here

    // draw trajectory in pangolin
    DrawTrajectory(poses);
    return 0;
}
```

输出轨迹：



这个题我做得挺坎坷：耗费了好长时间

使用第一个电脑画出来的图如下：



图片一直在抖，无法画完整全部，轨迹最后没有连接到蓝色部分。不知道为什么，我找了很久代码问题，觉得没问题。

最后我换了电脑尝试同样的代码，然后就顺利输出了。

有时候啊，这环境到底咋回事，真的是很难说清楚。按道理我的两个linux电脑的环境都一样，使用的pagolin和sophus包也是一样的。

六、轨迹的误差

利用提供的两个txt轨迹文件，实现RMSE的计算代码，给出最终结果。

回答：

二范数定义：谱范数，常用语度量距离。A^T*A的最大特征值的开平方。

```
string truth_file = "../groundtruth.txt";
string estimated_file = "../estimated.txt";
ifstream trajectory;
ifstream estimated_trajectory;
// 两个要做运算的变换矩阵
Sophus::SE3 Twc_g;
Sophus::SE3 Twc_e;

Eigen::Quaterniond q_g;
Eigen::Quaterniond q_e;

Eigen::Vector3d t_g;
Eigen::Vector3d t_e;

Eigen::Matrix<double ,6,1> temp;
```

```
double time_g;
double time_e;

double err = 0;
double RMSE = 0;

int main(int argc, char **argv) {

    trajectory.open(truth_file.c_str());
    if (!trajectory.is_open()){
        cout << "the file is empty!!" << endl;
        return -1;
    }

    estimated_trajectory.open(estimated_file.c_str());
    if (!estimated_trajectory.is_open()){
        cout << "the file is empty!!" << endl;
        return -1;
    }

    int num = 0;
    string sGLine, sELine;
    while(getline(trajectory, sGLine) && !sGLine.empty()
                    &&getline(estimated_trajectory, sELine) && !sELine.empty()){
        istringstream issG(sGLine);
        istringstream issE(sELine);
        issG >> time_g >> t_g[0] >> t_g[1] >> t_g[2] >>
q_g.x()>>q_g.y()>>q_g.z()>>q_g.w();
        issE >> time_e >> t_e[0] >> t_e[1] >> t_e[2] >>
q_e.x()>>q_e.y()>>q_e.z()>>q_e.w();

        Twc_e = Sophus::SE3(q_e,t_e);
        Twc_g = Sophus::SE3(q_g,t_g);

        // .log()表示转换为李代数，平移在前旋转在后
        temp = (Twc_g.inverse() * Twc_e).log();
        err += temp.transpose() * temp;
        num ++;
    }

    RMSE = sqrt(err/num);
    trajectory.close();
    estimated_trajectory.close();
    // end your code here

    cout << "RMSE:" << RMSE << endl;

    return 0;
}
```

程序运行结果

```
/home/xbot/src/MyWork/draw_trajectory
RMSE:2.20728

Process finished with exit code 0
```

将两条轨迹画在同一个图里，观察轨迹之间的误差。

回答：

```
string truth_file = "../groundtruth.txt";
string estimated_file = "../estimated.txt";

// function for plotting trajectory, don't edit this code
// start point is red and end point is blue
void DrawTrajectory(vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>>);

int main(int argc, char **argv) {

    vector<Sophus::SE3, Eigen::aligned_allocator<Sophus::SE3>> poses;

    /// implement pose reading code
    // start your code here (5~10 lines)
    Eigen::Quaterniond q;
    Eigen::Vector3d t;
    Sophus::SE3 T;
    ifstream trajectory;
    ifstream estimated_trajectory;
    double time_stamp;

    trajectory.open(truth_file.c_str());
    if (!trajectory.is_open()){
        cout << "the file is empty!!" << endl;
        return -1;
    }

    estimated_trajectory.open(estimated_file.c_str());
    if (!estimated_trajectory.is_open()){
        cout << "the file is empty!!" << endl;
        return -1;
    }

    string sLine;
```

```
    while(getline(trajectory, sLine) && !sLine.empty()){
        istringstream iss(sLine);
        iss >> time_stamp >> t[0] >> t[1] >> t[2] >> q.x() >> q.y() >> q.z() >>
q.w();
        T = Sophus::SE3(q, t);
        poses.push_back(T);
    }

    while(getline(estimated_trajectory, sLine) && !sLine.empty()){
        istringstream iss(sLine);
        iss >> time_stamp >> t[0] >> t[1] >> t[2] >> q.x() >> q.y() >> q.z() >>
q.w();
        T = Sophus::SE3(q, t);
        poses.push_back(T);
    }

    trajectory.close();
    estimated_trajectory.close();
    // end your code here

    DrawTrajectory(poses);
    return 0;
}
```