附录

1 Mathematica

main

```
(* 群的元素列表 *)
GroupElements[Group]

(* 验证元素是否数域该群 *)
GroupElementQ[Group, g]

(* 群的阶 *)
GroupOrder[Group]

(* n阶对称群 *)
SymmetricGroup[n]

(* 2n阶二面体群 *)
DihedralGroup[n]

(* n阶循环群 *)
CyclicGroup[n]

(* 定义模n群 *)
ModuloGroup[n_] := Range[1, n - 1]

(* 定义模n单位群 *)
ModuloUnitGroup[n_] := Select[Range[1, n - 1], GCD[#, n] == 1 &]

(* 定义判断模n单位群是否为循环群的函数 *)
IsCyclicModuloUnitGroup[n_] := Module[
  {primitiveRoots}, primitiveRoots = PrimitiveRootList[n];
  If[Length[primitiveRoots] == 0, False, True]
]

(* 计算模n单位群的生成元 *)
PrimitiveRootList[n]

(* 计算原根 *)
PrimitiveRootList[n]

```

2 MATLAB

定义名为“Group”的数据结构，包含`element`、`group`和`order`三个属性。

- `element`：群元素。每一行代表一个元素，第k行元素的指标定义为k。
- `group`：群运算表。(i,j)处的数字表示i指标对应元素与j指标对应元素作运算后的元素对应的指标。
- `order`：群的阶。

文件清单:

名称	类型
main	脚本
循环群	函数
模n单位群	函数
对称群	函数
交错群	函数
二面体群	函数
群运算	函数
群元素的逆	函数
群元素的阶	函数
判断是否可交换	函数
群直积	函数
子群	函数
正规子群	函数
换位子群	函数
中心子群	函数
共轭类	函数
中心化子	函数
群同态态射	函数
群同构态射	函数
Euler函数	函数
全排列	函数
全组合	函数

main

```
clear; clc

%% 一、常见群

% 1.n阶循环群(模n群)
[element, group, realGroup, order] = cyclicGroup(n);

% 2.模n单位群
[element, group, realGroup, order] = moduloNGroup(n);
```

```
% 3.n阶对称群
[element, group, order] = symmetricGroup(n);

% 4.n阶二面体群
[element, group, order] = dihedralGroup(n);

%% 二、群运算

% 1.群运算
result = groupOperate(Group, g, h);

% 2.群元素的逆
matrix = inverseMatrix(group);

% 3.群元素的阶
atrix = orderMatrix(group);

% 4.判断是否可交换
ifCommutative(group);

% 5.群直积
[element, group, order] = directProductOfGroups(G, H);

% 6.子群
subgroupMatrix = subgroup(group);

% 7.规范子群
normalSubgroupMatrix = normalSubgroup(group);

%% 三、范畴群

% 1.群同态态射
mapMatrix = groupHomomorphism(G, H);

% 2.群同构态射
mapMatrix = groupIsomorphism(G, H);

```

循环群

```
function [element, group, realGroup, order] = cyclicGroup(n)

    % 名称：模n循环群Z/nZ
    % 输入：循环群阶数n
    % 输出：1.群元素element；2.群运算表group和realGroup；3.群阶order
    % 说明：以element表中元素的位置建立与元素间的双射
    %       如下出现的数字均代表元素的位置，不代表真实元素
    %       realGroup为真实元素
    %       group(i,j)=element(i)+element(j)

    %% 1.群元素
    element = 0: n-1;

    %% 2.群阶
```

```
    order = n;

    %% 3.群运算表
    [i, j] = meshgrid(1: n, 1: n);
    realGroup = mod(i + j - 2, n);
    group = realGroup + 1;

end

```

模n单位群

```
function [element, group, realGroup, order] = moduloNGroup(n)

    % 名称：模n单位群(Z/nZ)*
    % 输入：模n单位群阶数n
    % 输出：1.群元素element；2.群运算表group和realGroup；3.群阶order
    % 说明：以element表中元素的位置建立与元素间的双射
    %       如下出现的数字均代表元素的位置，不代表真实元素
    %       realGroup为真实元素
    %       group(i,j)=element(i)*element(j)

    %% 1.群元素和群阶
    [order, element] = EulerFunction(n);

    %% 3.群运算表
    realGroup = mod(element' * element, n);
    group = zeros(order, order);
    for i = 1: order
        for j = 1: order
            for k = 1: order
                if element(k) == realGroup(i, j)
                    indice = k;
                    break
                end
            end
            % 赋值
            group(i, j) = indice;
        end
    end

end

```

对称群

```
function [element, group, order] = symmetricGroup(n)

    % 名称：n阶对称群S_n
    % 输入：对称群阶数n
    % 输出：1.群元素element；2.群运算表group；3.群阶order
    % 说明：以element表中元素的位置建立与元素间的双射
    %       如下出现的数字均代表元素的位置，不代表真实元素
    %       element为n!行n列矩阵

```

```
    %       其中element(k)代表第k个元素，即第k个双射
    %       第j列代表j在该行代表映射下的像
    %       group(i,j)=element(j) \circ element(i)

    %% 1.群元素
    element = fullPermutation(n);
    element = element(end: -1: 1, :);

    %% 2.群阶
    order = factorial(n);

    %% 3.群运算表

    % 初始化
    group = zeros(order, order);
    for i = 1: order
        for j = 1: order
            % 初始化
            permutation = zeros(1, n);
            permutation_i = element(i, :);
            permutation_j = element(j, :);
            % 计算复合映射
            for k = 1: n
                 permutation(k) = permutation_j(permutation_i(k));
            end
            % 找到复合映射的编号
            for k = 1: order
                if element(k, :) == permutation
                    indice = k;
                    break
                end
            end
            % 赋值
            group(i, j) = indice;
        end
    end

end

```

交错群

```
function [element, group, order] = alternatingGroup(n)

    % 名称：n阶交错群A_n
    % 输入：交错群阶数n
    % 输出：1.群元素element；2.群运算表group；3.群阶order
    % 说明：以element表中元素的位置建立与元素间的双射
    %       如下出现的数字均代表元素的位置，不代表真实元素
    %       element为n!行n列矩阵
    %       其中element(k)代表第k个元素，即第k个双射
    %       第j列代表j在该行代表映射下的像
    %       group(i,j)=element(j) \circ element(i)

    %% 1.群元素
```

```
    permutations = perms(1: n); % 生成所有n元置换的排列

    element = [];
    for i = 1:size(permutations, 1)
        perm = permutations(i, :);
        sign = 1; % 初始化符号为正号

        for j = 1:n
            for k = (j+1):n
                if perm(j) > perm(k)
                    sign = -sign; % 交换时改变符号
                end
            end
        end

        if sign == 1 % 符号为正号表示是偶置换
            element = [element; perm];
        end
    end
    element = element(end: -1: 1, :);

    %% 2.群阶
    order = factorial(n) / 2;

    %% 3.群运算表

    % 初始化
    group = zeros(order, order);
    for i = 1: order
        for j = 1: order
            % 初始化
            permutation = zeros(1, n);
            permutation_i = element(i, :);
            permutation_j = element(j, :);
            % 计算复合映射
            for k = 1: n
                 permutation(k) = permutation_j(permutation_i(k));
            end
            % 找到复合映射的编号
            for k = 1: order
                if element(k, :) == permutation
                    indice = k;
                    break
                end
            end
            % 赋值
            group(i, j) = indice;
        end
    end

end

```

[二面体群]

```m
function [element, group, order] = dihedralGroup(n)

    % 名称：n阶二面体群D_2n
    % 输入：二面体群阶数n
    % 输出：1.群元素element；2.群运算表group；3.群阶order
    % 说明：以element表中元素的位置建立与元素间的双射
    %       如下出现的数字均代表元素的位置，不代表真实元素
    %       element为2n行2列矩阵
    %       其中element(k)代表第k个元素
    %       element的行向量(i,j)代表映射sigma^i tau^j
    %       sigma^n = tau^2 = sigma tau sigma tau = 1
    %       group(i,j)=element(i) \circ element(j)

    %% 1.群元素
    element = [(0: n-1)', zeros(n, 1); (0: n-1)', ones(n, 1)];

    %% 2.群阶
    order = 2 * n;
    %% 3.群运算表

    % 初始化
    group = zeros(order, order);
    for i = 1: order
        for j = 1: order

            % 利用sigma^n = tau^2 = sigma tau sigma tau = 1降次
            temp = [element(i, :), element(j, :)];

            % tau^2 = sigma^n = 1降次
            if temp(2) == 0
                map = [mod(temp(1)+temp(3), n), temp(4)];
            elseif temp(3) == 0
                map = [temp(1), mod(temp(2)+temp(4), 2)];
            else
                map = [mod(n+temp(1)-temp(3), n), mod(1+temp(4), 2)];
            end

            % 找到复合映射的编号
            for k = 1: order
                if element(k, :) == map
                    indice = k;
                    break
                end
            end
            % 赋值
            group(i, j) = indice;

        end
    end

end

```

群运算

```
function result = groupOperate(Group, g, h)

    % 名称：群运算
    % 输入：群Group，以及群元素编号g和h
    % 输出：群元素Group(g)*Group(h)的编号
    % 关于群Group：
    % 给定群Group中元素的编号：1, ..., |Group|
    % Group(n)表示群G中编号为n的元素
    % Group为|G|行|G|列矩阵，其中Group(i,j)=Group(i)*Group(j)

    %% 计算群运算
    result = Group(g, h);

end

```

群元素的逆

```
function matrix = inverseMatrix(group)

    % 名称：逆元素矩阵
    % 输入：群运算表group
    % 输出：逆元素编号矩阵matrix
    % 说明：matrix(k)表示编号为k的元素的逆元素的编号

    %% 函数
    order = size(group, 1);
    matrix = zeros(1, order);
    for i = 1: order
        for j = 1: order
            if group(i, j) == 1
                matrix(i) = j;
                break
            end
        end
    end

end

```

群元素的阶

```
function matrix = orderMatrix(group)

    % 名称：元素阶矩阵
    % 输入：群运算表group
    % 输出：元素阶矩阵matrix
    % 说明：matrix(k)表示编号为k的元素的阶

    %% 函数
    order = size(group, 1);
    matrix = zeros(1, order);
    for loc = 1: order
        if loc == 1

```

```
            matrix(loc) = 1;
        else
            element = loc;
            for n = 2: order
                element = group(loc, element);
                if element == 1
                    matrix(loc) = n;
                end
            end
        end
    end

end

```

判断是否可交换

```
function judge = ifCommutative(G)

    % 名称：判断是否为交换群
    % 输入：群运算表
    % 输出：若可交换，则输出True；否则，输出False

    judge = all(all(G - G' == 0));

end

```

群直积

```
function [element, group, order] = directProductOfGroups(G, H)

    % 名称：群的直积
    % 输入：群G和H
    % 输出：1.群元素element；2.群运算表group；3.群阶order
    % 说明：以element表中元素的位置建立与元素间的双射
    %       如下出现的数字均代表元素的位置，不代表真实元素
    %       element为order行2列矩阵
    %       其中element(k)代表第k个元素
    %       element的行向量(i,j)代表G的第i个元素与H的第j个元素
    %       group(i,j)=element(i) * element(j)


    %% 1.群阶
    m = size(G, 1);
    n = size(H, 1);
    order = m * n;

    %% 2.群元素
    element = zeros(order, 2);
    for k = 1: order
        element(k, 1) = ceil(k / n);
        element(k, 2) = k - n * (ceil(k / n)-1);
    end

```

```
    %% 3.群运算表
    group = zeros(order, order);

    for i = 1: order
        for j = 1: order
            group(i, j) = ...
                n * (G(element(i, 1), element(j, 1)) - 1) ...
                + H(element(i, 2), element(j, 2));
        end
    end

end

```

子群

```
function subgroupMatrix = subgroup(group)

    % 名称：子群
    % 输入：群运算表group
    % 输出：所有子群
    % 说明：subgroupMatrix每一行代表一个子群的元素的编号

    %% 求解子群的阶
    order = size(group, 1);
    factors = [];
    loc = 1;
    for fac = 1: order
        if mod(order, fac) == 0
            factors(loc) = fac;
            loc = loc + 1;
        end
    end

    %% 求解子群
    invMatrix = inverseMatrix(group);
    subgroupMatrix = [];
    for n = factors
        combinations = generateCombinations(n, order);
        for com = combinations'
            judge = 1;
            for i = com'
                for j = com'
                    invj = invMatrix(j);
                    if ~ismember(group(i, invj), com)
                        judge = 0;
                        break
                    end
                end
                if judge == 0
                    break
                end
            end
            if judge == 1 && i == com(end) && j == com(end)
```

```
                subgroupMatrix = [subgroupMatrix; com', zeros(1, order - size(com', 2))];
            end
        end
    end

end

```

正规子群

```
function normalSubgroupMatrix = normalSubgroup(group)

    % 名称：正规子群
    % 输入：群运算表group
    % 输出：所有正规子群
    % 说明：normalSubgroupMatrix每一行代表一个正规子群的元素的编号

    %% 函数1
    order = size(group, 1);
    subgroupMatrix = subgroup(group);
    subgroupNumber = size(subgroupMatrix, 1);
    invmatrix = inverseMatrix(group);

    normalSubgroupMatrix = [];
    for N = 1: subgroupNumber

        judge = 1;
        normalSubgroup = subgroupMatrix(N, :);

        for n = 1: order

            if normalSubgroup(n) == 0
                break
            end

            for g = 1: order
                temp = group(group(g, normalSubgroup(n)), invmatrix(g));
                if ~ismember(temp, normalSubgroup)
                    judge = 0;
                    break
                end
            end

            if judge == 0
                break
            end
        end

        if judge == 1
            normalSubgroupMatrix = [normalSubgroupMatrix; normalSubgroup];
        end
    end

    %% 函数2
```

```
    % order = size(group, 1);
    % subgroupMatrix = subgroup(group);
    % subgroupNumber = size(subgroupMatrix, 1);
    %
    % normalSubgroupMatrix = [];
    % for N = 1: subgroupNumber
    %
    %     judge = 1;
    %     normalSubgroup = subgroupMatrix(N, :);
    %     normalSubgroup(normalSubgroup == 0) = [];
    %     normalSubgroupNumber = size(normalSubgroup, 2);
    %
    %     for g = 1: order
    %         gN = zeros(normalSubgroupNumber, 1);
    %         Ng = zeros(normalSubgroupNumber, 1);
    %         for n = 1: normalSubgroupNumber
    %             gN(n) = group(g, normalSubgroup(n));
    %             Ng(n) = group(normalSubgroup(n), g);
    %         end
    %         gN = unique(sort(gN));
    %         Ng = unique(sort(Ng));
    %         if ~isequal(gN, Ng)
    %             judge = 0;
    %             break
    %         end
    %     end
    %     if judge == 1
    %         normalSubgroupMatrix = [normalSubgroupMatrix; normalSubgroup,
zeros(1, order-size(normalSubgroup, 2))];
    %     end
    %
    % end

end

```

换位子群

```
function commutatorGroupMatrix = commutatorGroup(group)

    % 名称：换位子群
    % 输入：群运算表group
    % 输出：换位子群元素的编号

    %% 函数
    order = size(group, 1);
    matrix = inverseMatrix(group);
    commutatorGroupMatrix = zeros(1, order^2);
    for i = 1: order
        for j = 1: order
            commutatorGroupMatrix((i-1) * order + j) = group(group(group(i,
j), matrix(i)), matrix(j));
        end
    end
    commutatorGroupMatrix = sort(unique(commutatorGroupMatrix));

```

```
end

```

中心子群

```
function centreGroupMatrix = centerGroup(group)

    % 名称：中心子群
    % 输入：群运算表group
    % 输出：中心子群元素的编号

    %% 函数
    centreGroupMatrix = [];
    order = size(group, 1);
    for n = 1: order
        judge = 1;
        for k = 1: order
            if group(n, k) ~= group(k, n)
                judge = 0;
            end
        end
        if judge == 1
            centreGroupMatrix = [centreGroupMatrix, n];
        end
    end

end

```

共轭类

```
function class = conjugacyClass(g, group)

    % 名称：共轭类
    % 输入：元素编号g，群运算表group
    % 输出：编号为g的元素的共轭类的编号

    %% 函数
    order = size(group, 1);
    matrix = inverseMatrix(group);
    class = [];
    for n = 1: order
        element = group(group(n, g), matrix(n));
        class = [class, element];
    end
    class = sort(unique(class));

end

```

中心化子

```
function matrix = centralizer(g, group)
```

```
    % 名称：中心化子
    % 输入：元素编号g，群运算表group
    % 输出：编号为g的元素的共轭类的编号

    %% 函数
    order = size(group, 1);
    matrix = [];
    for n = 1: order
        if group(g, n) == group(n, g)
            matrix = [matrix, n];
        end
    end

end

```

群同态态射

```
function mapMatrix = groupHomomorphism(G, H)

    % 名称：群同态态射
    % 输入：群G和H
    % 输出：G-->H的同态态射
    % 其中每一行代表一个态射，第j列代表j在该态射下的像

    %% 函数

    % 定义群的阶
    m = size(G, 1);
    n = size(H, 1);

    % 初始化映射矩阵
    mapMatrix = zeros(n^m, m);

    % 生成所有可能的映射
    for i = 1: n^m
        temp = i - 1;
        for j = m:-1:1
            quotient = floor(temp / n);
            remainder = mod(temp, n);
            mapMatrix(i, j) = remainder + 1;
            temp = quotient;
        end
    end

    % 初始化元素矩阵
    elementMatrix = [];

    % 生成元素矩阵
    for k = 1: m
        elementMatrix = [elementMatrix, [k*ones(1, m); 1: m]];
    end

    % 筛选同态态射
```

```
    for k = 1: n^m
        for element = elementMatrix
            if mapMatrix(k, groupOperate(G, element(1), element(2))) ~=
groupOperate(H, mapMatrix(k, element(1)), mapMatrix(k, element(2)))
                mapMatrix(k, :) = zeros(1, m);
                break
            end
        end
    end

    % 删除非同态态射
    mapMatrix = mapMatrix(any(mapMatrix, 2), :);

end

```

群同构态射

```
function mapMatrix = groupIsomorphism(G, H)

    % 名称：群同构态射
    % 输入：群G和H
    % 输出：G-->H的同构态射
    % 其中每一行代表一个态射，第j列代表j在该态射下的像

    %% 函数

    % 定义群的阶
    m = size(G, 1);
    n = size(H, 1);

    if m == n

        % 计算G-->H的同态态射
        mapMatrix = groupHomomorphism(G, H);

        % 定义目标排列
        permutation = 1: m;

        % 初始化一个逻辑向量，用于标记符合条件的行
        is_permutation = false(size(mapMatrix, 1), 1);

        % 遍历每一行
        for k = 1: size(mapMatrix, 1)
            % 判断是否是目标排列
            if isequal(sort(mapMatrix(k, :)), permutation)
                is_permutation(k) = true;
            end
        end

        % 从矩阵中选择符合条件的行
        mapMatrix = mapMatrix(is_permutation, :);

    else
        mapMatrix = [];
```

```
    end

end

```

Euler函数

```
function [N, Matrix] = EulerFunction(n)

    % 生成 1 到 n-1 的矩阵
    numbers = 1: n-1;

    % 计算每个数与 n 的最大公约数
    gcdMatrix = gcd(numbers, n);

    % 使用逻辑索引找到与 n 互素的数
    coprime = numbers(gcdMatrix == 1);

    % 将结果转换为矩阵
    Matrix = reshape(coprime, 1, []);

    % 计算元素个数
    N = size(Matrix, 2);

end

```

全排列

```
function permutation = fullPermutation(n)

    % 名称：全排列
    % 输入：n
    % 输出：n!行n列矩阵，每一行为一个全排列

    %%

    if n == 1
        permutation = 1;
    else
        sub_permutations = fullPermutation(n - 1);
        m = size(sub_permutations, 1);
        permutation = zeros(m * n, n);
        for i = 1: m
            for j = 1: n
                permutation((i - 1) * n + j, :) = [sub_permutations(i, 1:j-1), n, sub_permutations(i, j:end)];
            end
        end
    end

end

```

全组合

```
function combinations = generateCombinations(k, n)
    % 生成组合矩阵的函数
    % 输入：组合大小k，元素总数n
    % 输出:一个C_n^k行，k列矩阵，包含所有可能的组合

    %% 函数

    % 初始化组合矩阵为空
    combinations = [];
    % 初始化当前组合为空
    currentCombination = [];
    % 调用递归函数来生成组合
    generate(1, k, n, currentCombination);

    % 递归函数，生成所有可能的组合
    % 输入参数:
    %   - start: 当前迭代的起始元素
    %   - k: 剩余要选择的元素数量
    %   - n: 候选元素的总数
    %   - currentCombination: 当前的组合
    function generate(start, k, n, currentCombination)
        % 当剩余要选择的元素数量为0时，表示已经生成一个组合
        if k == 0
            % 将当前组合添加到组合矩阵中
            combinations = [combinations; currentCombination];
            return;
        end

        % 遍历候选元素，生成组合
        for i = start:n
            % 递归调用generate函数，继续生成组合
            generate(i+1, k-1, n, [currentCombination, i]);
        end
    end

end

```