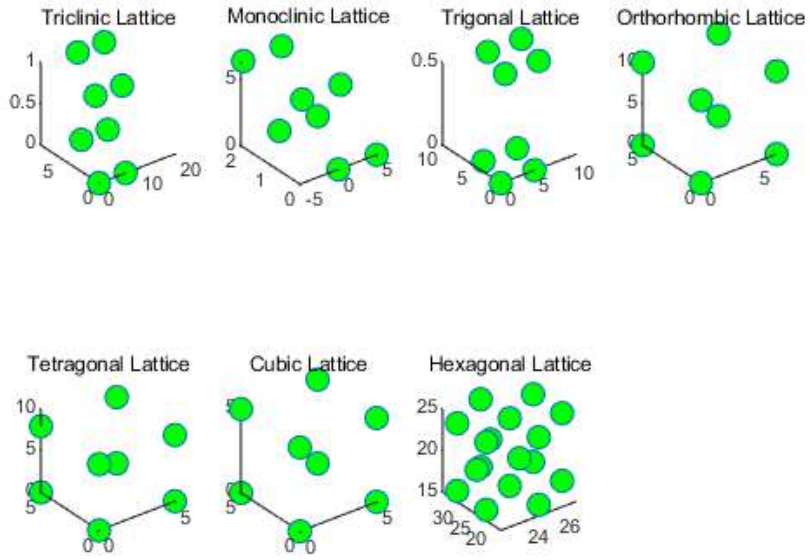
```
% Open source on 开源于
% https://github.com/Techy-Wu/MATLAB-learning/tree/8f7d3a6a3edda5ad190737fc7fd204305d90a95e/7%E5%A4%A7%E6%99%B6%E7%B3%BB

% 初始化工作区
clc;
clear;

% 以下7个晶系均按照6行代码实现，具体功能如下：
% 1 - 晶体类型标记（注释行）
% 2 - 调用函数建立晶系
% 3 - 设定并划分MATLAB作图区
% 4 - MATLAB作图
% 5 - 添加MATLAB作图标题
% 6 - 立方化MATLAB作图坐标
% 7 - 调用函数输出至PDB文件
%

% Triclinic Lattice
lattice = build_lattice('Triclinic', 1, 'a', 7.0, 'b', 10.0, 'c', 2.0, 'alpha', 0.8 * pi, 'beta', 0.7 * pi, 'gamma', 0.2 * pi);
subplot(2, 4, 1);
plot3(lattice(:, 1), lattice(:, 2), lattice(:, 3), 'o', 'MarkerFaceColor','g', 'MarkerSize', 10);
title('Triclinic Lattice');
axis square;
damp_to_rasmol('Triclinic Lattice.pdb', lattice, 'Cu');

% Monoclinic Lattice
lattice = build_lattice('Monoclinic', 1, 'a', 5.0, 'b', 2.0, 'c', 8.0, 'beta', 0.7 * pi);
subplot(2, 4, 2);
plot3(lattice(:, 1), lattice(:, 2), lattice(:, 3), 'o', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
title('Monoclinic Lattice');
axis square;
damp_to_rasmol('Monoclinic Lattice.pdb', lattice, 'Cu');

% Trigonal Lattice
lattice = build_lattice('Trigonal', 1, 'a', 5.0, 'alpha', 0.4 * pi);
subplot(2, 4, 3);
plot3(lattice(:, 1), lattice(:, 2), lattice(:, 3), 'o', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
title('Trigonal Lattice');
axis square;
damp_to_rasmol('Trigonal Lattice.pdb', lattice, 'Cu');

% Orthorhombic Lattice
lattice = build_lattice('Orthorhombic', 1, 'a', 7.0, 'b', 5.0, 'c', 10.0);
subplot(2, 4, 4);
plot3(lattice(:, 1), lattice(:, 2), lattice(:, 3), 'o', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
title('Orthorhombic Lattice');
axis square;
damp_to_rasmol('Orthorhombic Lattice.pdb', lattice, 'Cu');

% Tetragonal Lattice
lattice = build_lattice('Tetragonal', 1, 'a', 5.0, 'c', 8.0);
subplot(2, 4, 5);
plot3(lattice(:, 1), lattice(:, 2), lattice(:, 3), 'o', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
title('Tetragonal Lattice');
axis square;
damp_to_rasmol('Tetragonal Lattice.pdb', lattice, 'Cu');

% Cubic Lattice
lattice = build_lattice('Cubic', 1, 'a', 5.0);
subplot(2, 4, 6);
plot3(lattice(:, 1), lattice(:, 2), lattice(:, 3), 'o', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
title('Cubic Lattice');
axis square;
damp_to_rasmol('Cubic Lattice.pdb', lattice, 'Cu');

% Hexagonal Lattice
lattice = build_lattice('Hexagonal', 5, 'a', 5.0);
% 使用截取工具截取合适区域
lattice = lattice_slicer(lattice, [22, 17, 16], [22, 17, 16] + [10, 18, 9]); % 具体边界数据手动计算得到
% 修剪数据，去除不属于HCP基础晶格的原子
```

```
lattice = [lattice(1:16, :); lattice(18, :)];
subplot(2, 4, 7);
plot3(lattice(:, 1), lattice(:, 2), lattice(:, 3), 'o', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
title('Hexagonal Lattice');
axis square;
damp_to_rasmol('Hexagonal Lattice.pdb', lattice, 'Cu');

% Function Definition
% 晶系构造函数
function [lattice] = build_lattice(lattice_name, enlarge_limit, varargin) % MATLAB中varargin标记标明函数接受可变长度形参元胞
    % 校验扩增限制
    if enlarge_limit < 0
        % 扩增限制为负抛出异常
        error('Invalid enlarge limitation');
    end
    % 分支不同类型晶系
    switch lattice_name
        % 以三斜晶系为例标注代码解释
        case 'Triclinic'
            %获取主要参数
            try
                a = get_parameter('a');
                b = get_parameter('b');
                c = get_parameter('c');
                alpha = get_parameter('alpha');
                beta = get_parameter('beta');
                gamma = get_parameter('gamma');
            catch
                % 获取失败抛出异常
                error('Parameters invalid or not enough');
            end
            % 初始化主要参数
            % 晶胞原子
            atom1 = [0.0, 0.0, 0.0];
            % 基矢量
            ux = [a, 0.0, 0.0];
            uy = [b * cos(gamma), b * sin(gamma), 0.0];
            uz = [c * sin(beta), c * sin(alpha), c * cos(beta) * cos(alpha)];
            % 扩增法建立晶系结构
            id = 1;
            for i = 0:enlarge_limit
                for j = 0:enlarge_limit
                    for k = 0:enlarge_limit
                        % 确定本次扩增的移动矢量
                        vector = ux * i + uy * j + uz * k;
                        % 添加原子
                        lattice(id, 1:3) = atom1 + vector;
                        id = id + 1;
                    end
                end
            end

        case 'Monoclinic'
            try
                a = get_parameter('a');
                b = get_parameter('b');
                c = get_parameter('c');
                beta = 0.7*pi;
            catch
                error('Parameters invalid or not enough');
            end
            atom1 = [0.0, 0.0, 0.0];
            ux = [a, 0.0, 0.0];
            uy = [0.0, b, 0.0];
            uz = [c * cos(beta), 0.0, c * sin(beta)];
            id = 1;
            for i = 0:enlarge_limit
                for j = 0:enlarge_limit
                    for k = 0:enlarge_limit
                        vector = ux * i + uy * j + uz * k;
                        lattice(id, 1:3) = atom1 + vector;
```

```
                id = id + 1;
            end
        end
    end

case 'Trigonal'
    try
        a = get_parameter('a');
        alpha = get_parameter('alpha');
    catch
        error('Parameters invalid or not enough');
    end
    atom1 = [0.0, 0.0, 0.0];
    ux = [a, 0.0, 0.0];
    uy = [a * cos(alpha), a * sin(alpha), 0.0];
    uz = [a * sin(alpha), a * sin(alpha), a * cos(alpha) * cos(alpha)];
    id = 1;
    for i = 0:enlarge_limit
        for j = 0:enlarge_limit
            for k = 0:enlarge_limit
                vector = ux * i + uy * j + uz * k;
                lattice(id, 1:3) = atom1 + vector;
                id = id + 1;
            end
        end
    end

case 'Orthorhombic'
    try
        a = get_parameter('a');
        b = get_parameter('b');
        c = get_parameter('c');
    catch
        error('Parameters invalid or not enough')
    end
    atom1 = [0.0, 0.0, 0.0];
    ux = [a, 0.0, 0.0];
    uy = [0.0, b, 0.0];
    uz = [0.0, 0.0, c];
    id = 1;
    for i = 0:enlarge_limit
        for j = 0:enlarge_limit
            for k = 0:enlarge_limit
                vector = ux * i + uy * j + uz * k;
                lattice(id, 1:3) = atom1 + vector;
                id = id + 1;
            end
        end
    end

case 'Tetragonal'
    try
        a = get_parameter('a');
        c = get_parameter('c');
    catch
        error('Parameters invalid or not enough');
    end
    atom1 = [0.0, 0.0, 0.0];
    ux = [a, 0.0, 0.0];
    uy = [0.0, a, 0.0];
    uz = [0.0, 0.0, c];
    id = 1;
    for i = 0:enlarge_limit
        for j = 0:enlarge_limit
            for k = 0:enlarge_limit
                vector = ux * i + uy * j + uz * k;
                lattice(id, 1:3) = atom1 + vector;
                id = id + 1;
            end
        end
    end
```

```
        case 'Cubic'
            try
                a = get_parameter('a');
            catch
                error('Parameters invalid or not enough');
            end
            atom1 = [0.0, 0.0, 0.0];
            ux = [a, 0.0, 0.0];
            uy = [0.0, a, 0.0];
            uz = [0.0, 0.0, a];
            id = 1;
            for i = 0:enlarge_limit
                for j = 0:enlarge_limit
                    for k = 0:enlarge_limit
                        vector = ux * i + uy * j + uz * k;
                        lattice(id, 1:3) = atom1 + vector;
                        id = id + 1;
                    end
                end
            end

        case 'Hexagonal'
            try
                a = get_parameter('a');
                try
                    c = get_parameter('c');
                catch
                    c = a * 1.633;
                end
            catch
                error('Parameters invalid or not enough');
            end
            atom1 = [0.0, 0.0, 0.0];
            atom2 = [0.5 * a, sqrt(3) / 2 * a, 0.0];
            atom3 = [0.0, 2. / sqrt(3) * a, 0.5 * c];
            atom4 = [-0.5 * a, -a / sqrt(3) / 2, 0.5 * c];
            ux = [a, 0.0, 0.0];
            uy = [0.0, sqrt(3) * a, 0.0];
            uz = [0.0, 0.0, c];
            id = 1;
            for i = 0:enlarge_limit
                for j = 0:enlarge_limit
                    for k = 0:enlarge_limit
                        vector = ux * i + uy * j + uz * k;
                        lattice(id, 1:3) = atom1 + vector;
                        id = id + 1;
                        lattice(id, 1:3) = atom2 + vector;
                        id = id + 1;
                        lattice(id, 1:3) = atom3 + vector;
                        id = id + 1;
                        lattice(id, 1:3) = atom4 + vector;
                        id = id + 1;
                    end
                end
            end

        otherwise
            % 晶格名称错误抛出异常
            error('Invalid lattice name');
    end

    % 参数获取函数
    function result = get_parameter(parameter_name)
        % 参数组中查找并返回指定参数
        result = varargin{find(strcmp(parameter_name, varargin)) + 1};
    end
end

% 晶格裁切函数
function output_lattice = lattice_slicer(input_lattice, min_limit, max_limit)
```

```
    id = 1;
    for i = 1:length(input_lattice)
        if sum(input_lattice(i, 1:3) >= min_limit) == 3 && sum(input_lattice(i, 1:3) <= max_limit) == 3
            % 注：MATLAB中矩阵的逻辑运算将返回矩阵形式结果，矩阵全为1即全为TRUE
            output_lattice(id, 1:3) = input_lattice(i, 1:3);
            id = id + 1;
        end
    end
end

% PDB文件输出函数
function damp_to_rasmol(file_name, lattice, element)
    % 打开文件并存储文件句柄
    file_ID = fopen(file_name, 'w');
    % 定义rasmol可接受的文件格式范例
    rasmol_format_spec = '%-6s%-5d %-4s%-1s%-3s  %-4d%-1s    %-8.3f%-8.3f%-8.3f\n';
    % 遍历输出各个原子
    num = 0;
    for id = 1:length(lattice)
        num = num + 1;
        fprintf(file_ID, rasmol_format_spec, 'ATOM', num, element, ' ', ' ', 1, ' ', lattice(id, 1), lattice(id, 2), lattice(id, 3));
    end
    % 关闭文件
    fclose(file_ID);
end
```



Published with MATLAB® R2023b