1 问题描述

本次程序主要为了实现自动配平化学方程式，对于高中涉及的简单方程式，尚可手动配平，但是对于某些化学式复杂的有机方程式，手动配平则不太现实。

2 问题分析

实现过程其实很简单，首先将输入物输出物按照化学式分割开，再对每一个化学式按照元素分割开，并将其下标与元素一一对应。例如：

对$Al_2(SO_4)_3+NH_3\cdot H_2O\rightarrow Al(OH)_3+(NH_4)_2SO_4$，首先按照每个化学式分开成一个个字符串，然后对每个字符串再进行分割，例如$Al_2(SO_4)_3$，首先确定括号的位置，把括号内的元素及其下标(包括括号外的下标)单独拿出($(SO_4)_3$)，对剩余元素(Al_2)重新遍历。遍历到大写字母就停止，然后判断其下一个字符是什么：如果是小写字母，说明大写字母与小写字母组合是一个元素，再判断再下一位是什么，如果是数字就是这个元素的下标，如果是大写字母说明他的下标为1；如果是数字，那么说明这个大写字母就是元素，数字是他的下标；如果是大写字母，说明这个元素下标为1。对括号内的元素也相同处理，只是需要最后将下标乘上括号之外的下标才能得到元素数量。最后一步，将所有重复出现的元素合并，得到两个数组，第一个存放着这个化学式含有的元素，第二个存放着化学式中每个元素对应的数量。

然后给每个化学式赋值系数：$aAl_2(SO_4)_3+bNH_3\cdot H_2O\rightarrow cAl(OH)_3+d(NH_4)_2SO_4$，按照元素守恒，可以得出线性方程组。

$$\begin{cases}2a=c\\3a=d\\12a+b=3c+4d\\b=2d\\5b=3c+8d\end{cases}$$

可以发现得到的方程组有如下特点：①必定是齐次的，②不一定为方阵，可能未知量比方程数量多，也可能未知量比方程数量少，③由于化学方程式自身的配平系数可以同比例放缩，因此方程组的解有无限个

既然方程组的解有无限个，那么系数矩阵的秩必定小于未知量个数，即使方程数量多，但是经过高斯消元化简，肯定有冲突的方程消去。因为非方阵的通解解法课程中没有学过，因此我们把 **未知量个数-系数矩阵秩** 数量的未知量进行赋值，一般赋值为1 ，然后重新整理方程组，这样一定会得到一个方阵，且是非齐次的。这样就可以用到我们学过的知识来求解这个方程组。最后得到的结果要乘最小公倍数使得每个数都为整数。

3 代码实现

3.1 数据预处理

还是以$Al_2(SO_4)_3+NH_3\cdot H_2O\rightarrow Al(OH)_3+(NH_4)_2SO_4$为例，假设输入为"Al2(S04)3,NH3·H2O"

```
strinput='Al2(SO4)3,NH3·H2O,H2O';
stroutput='Al(OH)3,(NH4)2SO4';
cheminput=strsplit(strinput,",");
chemoutput=strsplit(stroutput,",");
```

首先要按照英文逗号来把各化学式分开。然后对一个化学式，先不考虑括号，进行分解，要求输出两个数组，一个为包含的元素，一个为各元素对应的个数，两者是一一对应的。

实现思路为：首先遍历所有字符，只有是大写字母时才会进行判断。如果下一个是大写字母，那么他是元素，数量为1；如果下一个是小写字母，那么这两个字母合起来为元素，再看再下一个是什么来确定是否有数字；如果下一个为数字，那么从此位开始遍历，重新定义一个存放数字的数组，把遍历到的数字存放进去，再转化为数字作为该元素的数量。这其中涉及下一位的操作都要加一个是否存在下一位的判断，防止索引溢出。最后对存储元素及其数量的临时变量，把重复的元素合并，存到输出变量中。

同时还要注意，在matlab中关于字符串有两种，一个是单引号的字符数组char，一个是双引号的字符串string，在遍历每个字符时一定要先将字符串转化为字符数组char。

由此思路定义unpack_element函数：

```
function [element,element_num]=unpack_element(str)
    element_temp=[];
    num_temp=[];
    element=[];
    element_num=[];
    temp1=strrep(str,"·","");
    temp=char(temp1);
    for j=1:length(temp)%遍历每一个字符
        if (double(temp(j))>=65)&&(double(temp(j))<=90)%只有这个字符是大写字母时才会停
下
            if j==length(temp)%判断是否是最后一个字符
                element_temp=[element_temp,string(temp(j:j))];
                num_temp=[num_temp,1];
            else
                if (double(temp(j+1))>=97)&&(double(temp(j+1))<=122)%如果下一个是小
写字母
                    element_temp=[element_temp,string(temp(j:j+1))];%这两个字母组合
为元素名称
                    if j+1==length(temp)%判断接下来还有没有字符
                        num_temp=[num_temp,1];
                    else
                        if (double(temp(j+2))>=65)&&(double(temp(j+2))<=90)%如果再
下一个是大写字母
                            num_temp=[num_temp,1];%此元素下标为1
                        end
                        if (double(temp(j+2))>=48)&&(double(temp(j+2))<=57)%如果再
下一个是数字
                              k=j+2;
                              num=[];
                              while k<=length(temp)&&(double(temp(k))>=48)&&
(double(temp(k))<=57)%把下面所有数字位组合成字符串
                                num=[num,temp(k)];
                                k=k+1;
                              end
                              n=str2num(num);%字符串转数字即为下标
                              num_temp=[num_temp,n];
                        end
                    end
                end

                if (double(temp(j+1))>=65)&&(double(temp(j+1))<=90)%如果下一个是大写
字母
                    element_temp=[element_temp,string(temp(j:j))];
```

```
                    num_temp=[num_temp,1];
                end
                if (double(temp(j+1))>=48)&&(double(temp(j+1))<=57)%如果下一个是数字
                    element_temp=[element_temp,string(temp(j:j))];
                    k=j+1;
                    num=[];
                    while k<=length(temp)&&((double(temp(k))>=48)&&
(double(temp(k))<=57))%把下面所有数字位组合成字符串
                            num=[num,temp(k)];
                            k=k+1;
                    end
                    n=str2num(num);%字符串转数字即为下标
                    num_temp=[num_temp,n];
                end
            end
        end
    end
    if ~isempty(element_temp)
    element=[element,element_temp(1)];
    element_num=[element_num,num_temp(1)];
    if length(element_temp)>1
        for i=2:length(element_temp)%把相同元素合并起来
            flag=0;
            for j=1:length(element)%用element_temp里的值遍历element里的值，有就相加
数，没有就添上
                if element_temp(i)==element(j)
                    element_num(j)=element_num(j)+num_temp(i);
                    flag=1;
                    break;
                end
            end
            if flag==0
                element=[element,element_temp(i)];
                element_num=[element_num,num_temp(i)];
            end
        end
    end
    end
end
```

以上的函数能够实现不带括号的化学式的分解，因此我们再考虑上带括号的化学式时，可以先尝试将每个括号包括其下标提出去，剩下部分再组合成一个化学式，对这些部分再分别单独用上述的函数分隔开，最后把每部分得到的元素及其数量再合并。得到该化学式的元素分解。

具体思路为：遍历每个字符，把所有左括号"("和右括号")"的下标分别存储起来，其中每对左右括号的是一一对应的。再判断每个右括号后面是否带有数字，数字存入一个变量，作为括号内容的下标，数字的长度也要存起来。然后再对整体减去括号内容物以及括号后的下标，得到化学式，最后把各部分都分别拆解再组合。

代码如下：

```
function [element,element_num]=unpack_complex_chem(str)
    left=[];
    right=[];
    temp1=strrep(str,"·","");
    temp=char(temp1);
```

```
    element_with=[];
    element_with_temp=[];%存放括号里面的字符串
    num_with_temp=[];%存放每个括号的下标
    num_with_length=[];%存放每个括号下标数字长度，没有则为0
    for i=1:length(temp)%首先找到所有括号的位置
        if temp(i)=='('
            left=[left,i];
        end
        if temp(i)==')'
            right=[right,i];
        end
    end
    if ~isempty(left)%如果存在括号
        for i=1:length(left)
            element_with_temp=
[element_with_temp,string(temp(left(i)+1:right(i)-1))];%把括号内部的字符串提取出来
            if right(i)==length(temp)
                num_with_temp=[num_with_temp,1];
                num_with_length=[num_with_length,0];
            else
                k=right(i)+1;
                num=[];
                while k<=length(temp)&&((double(temp(k))>=48)&&(double(temp(k))
<=57))%对括号后的数字进行处理
                    num=[num,temp(k)];
                    k=k+1;
                end
                if isempty(num)%如果没有数字，下标1，长度0
                    num_with_temp=[num_with_temp,1];
                    num_with_length=[num_with_length,length(num)];
                else
                    n=str2num(num);%字符串转数字即为下标
                    num_with_temp=[num_with_temp,n];
                    num_with_length=[num_with_length,length(num)];
                end
            end
        end
        str_temp=temp;
        for i=1:length(left)
            substring=temp(left(i):(right(i)+num_with_length(i)));
            str_temp=strrep(str_temp,substring,"");
        end
        element_with=[];
        element_with_num=[];
        for i=1:length(element_with_temp)
            [a,b]=unpack_element(element_with_temp(i));
            b=b.*num_with_temp(i);
            element_with=[element_with,a];
            element_with_num=[element_with_num,b];
        end
        [a,b]=unpack_element(str_temp);
        element_with=[element_with,a];
        element_with_num=[element_with_num,b];
    end
    if isempty(left)
        [element_with,element_with_num]=unpack_element(temp);%如果没有括号
```

```
        element=[];
        element_num=[];
    end
    if ~isempty(element_with)
    element=[];
    element_num=[];
    element=[element,element_with(1)];
    element_num=[element_num,element_with_num(1)];
    if length(element_with)>1
        for i=2:length(element_with)%把相同元素合并起来
            flag=0;
            for j=1:length(element)%用element_temp里的值遍历element里的值，有就相加
数，没有就添上
                if element_with(i)==element(j)
                    element_num(j)=element_num(j)+element_with_num(i);
                    flag=1;
                    break;
                end
            end
            if flag==0
                element=[element,element_with(i)];
                element_num=[element_num,element_with_num(i)];
            end
        end
    end
    end
end
```

下一步定义一个元胞数组，分别把输入输出涉及的化学式拆解后的元素及数量都存放的元胞数组中。

```
ele_matrix_input={};%定义元胞数组
elenum_matrix_input={};
for i=1:length(cheminput)
    [a,b]=unpack_complex_chem(cheminput(i));
    ele_matrix_input{end+1}=a;
    elenum_matrix_input{end+1}=b;
end
ele_matrix_output={};%定义元胞数组
elenum_matrix_output={};
for i=1:length(chemoutput)
    [a,b]=unpack_complex_chem(chemoutput(i));
    ele_matrix_output{end+1}=a;
    elenum_matrix_output{end+1}=b;
end
```

输出为

```
{["S"    "O"    "Al"]}    {["N"    "H"    "O"]}

{[3 12 2]}    {[1 5 1]}

{["O"    "H"    "Al"]}    {["N"    "H"    "S"    "O"]}

{[3 3 1]}    {[2 8 1 4]}
```

3.2 建立矩阵

首先确定参与反应的元素数量，根据元素守恒列方程式，其为方程式个数，即矩阵行数

```
a_marix=[];
a_marix=[a_marix,ele_matrix_input{1}(1)];
for i=1:length(ele_matrix_input)
    for j=1:length(ele_matrix_input{i})
        flag=0;
        for k=1:length(a_marix)
            if a_marix(k)==ele_matrix_input{i}(j)
                flag=1;
                break;
            end
        end
        if flag==0
            a_marix=[a_marix,ele_matrix_input{i}(j)];
        end
    end
end
```

然后确定化学式个数，即未知量个数，为矩阵的列数

```
a=length(a_marix);%参与反应的元素个数，根据元素守恒列方程式，为方程式个数，即矩阵行数
b=length(cheminput)+length(chemoutput);%化学式个数，即未知量个数，为矩阵的列数
```

然后定义一个(a×b)的零矩阵。按照a_marix中的元素顺序列方程组并填充数组。

```
A_temp=zeros(a,b);
for i=1:length(a_marix)
    for j=1:length(cheminput)
        flag=0;
        for k=1:length(ele_matrix_input{j})
            if ele_matrix_input{j}(k)==a_marix(i)
                A(i,j)=elenum_matrix_input{j}(k);
                flag=1;
                break;
            end
        end
        if flag==0
            A(i,j)=0;
        end
    end
    for j=length(cheminput)+1:b
        flag=0;
        l=j-length(cheminput);
        for k=1:length(ele_matrix_output{l})
            if ele_matrix_output{l}(k)==a_marix(i)
                A(i,j)=-elenum_matrix_output{l}(k);
                flag=1;
                break;
            end
        end
        if flag==0
```

```
            A(i,j)=0;
        end
    end
end
```

输出为

```
  3     0     0    -1
 12     1    -3    -4
  2     0    -1     0
  0     1     0    -2
  0     5    -3    -8
```

与我们之前得到的方程式一致。至于常数项，则全为0，是齐次方程组。

然后要构建方阵，根据化学方程组配平系数有无穷个，其秩肯定小于未知量个数。

对上述我们得到的矩阵，求得秩为3，说明其中至少有两行是与其他线性相关的，可以删去，只留下线性无关的三行。

具体删去哪几行，因为按顺序遍历$C_5^2=10$，最多要尝试十次，而且当数变更大时，程序将很难写，而且时间复杂度较大。因此我们使用随机数，随机删去两行直到剩余部分的秩为原矩阵的秩为止。

```
delete_num=a-rank(A);
A_temp=A;
A_temp(1,:)=[];
A_temp(2,:)=[];
while rank(A_temp)~=rank(A)
    A_temp=A;
    randomint=randperm(a,delete_num);
    A_temp(randomint,:)=[];
end
```

输出为

```
 12     1    -3    -4
  0     1     0    -2
  0     5    -3    -8
```

接下来构建方阵，需要先给 b(未知数个数)-rank(A) 个未知数赋初值1，然后便可转化为非齐次方阵求解问题。

```
give_num=b-rank(A);
b=[];
for i=1:rank(A)
    sum=0;
    for j=1:give_num
        sum=sum+A_temp(i,j);
    end
    b=[b,-sum];
end
A_temp(:,1:give_num)=[];
```

输出为：

```
    1     -3     -4
    1      0     -2
    5     -3     -8

  -12      0      0
```

3.3 解方程组

自此准备工作全部完成，进入解方程组的正题。
对于配平方程组的背景，我们应该尽量解能够是整数，或者说能够简单的转化为整数。

我们预定用三种方法求解

3.3.1 高斯消元法

```
function x=Gauss(a,b)
    [m,n]=size(a);
    for k=1:n-1
        for i=k+1:n
            factor=a(i,k)/a(k,k);
            for j=k+1:n
                a(i,j)=a(i,j)-factor*a(k,j);
            end
            b(i)=b(i)-factor*b(k);
        end
    end
    x(n)=b(n)/a(n,n);
    for i=n-1:-1:1
        sum=b(i);
       for j=i+1:n
           sum=sum-a(i,j)*x(j);
       end
       x(i)=sum/a(i,i);
    end
end
```

此为原始高斯消元，运算速度虽然有点慢，但是结果一定是好的。

```
function x=ColMajorGauss(a,b)
    [m,n]=size(a);
    for k=1:n-1
        col=a(k:n,k);%取出第k列对应的行值
        col=abs(col);%取绝对值
        [max_value,max_index]=max(col);%获得最大值的索引并+k-1转为真实索引
        if max_index+k-1 ~=k%对比是否为当前行，如果不是则换行
            a([k,max_index+k-1], :) = a([max_index+k-1,k],:);
            b([k,max_index+k-1])=b([max_index+k-1,k]);%常数项也要交换
        end
        for i=k+1:n%选好主元后再进行高斯消元
            factor=a(i,k)/a(k,k);
            for j=k:n
```

```
                a(i,j)=a(i,j)-factor*a(k,j);
            end
            b(i)=b(i)-factor*b(k);
        end
    end
    x(n)=b(n)/a(n,n);
    for i=n-1:-1:1
        sum=b(i);
       for j=i+1:n
           sum=sum-a(i,j)*x(j);
       end
       x(i)=sum/a(i,i);
    end
end
```

此为列主元消元，加上了一个交换列主元的方法，避免了病态方程组的危害。

3.3.2 LU分解

1，先求U第一行与L第一列：U的第一行$u_{1i}=a_{1i}$；L的第一列$l_{11}=1, l_{k1}=a_{k1}/u_{11}$

2，求出U前k-1行与L前k-1列后，第k步中计算U的第k行与L的第k列

$$u_{ki}=a_{ki}-\sum_{j=1}^{k-1}l_{kj}u_{ji} \qquad i=k,\ldots,n$$

$$l_{ik}=(a_{ik}-\sum_{j=1}^{k-1}l_{ij}u_{jk})/u_{kk} \qquad i=k+1,\ldots,n; k\neq n$$

向前代入求解LY=B

$$y_1=b_1$$

$$y_k=b_k-\sum_{j=1}^{k-1}l_{kj}y_j \qquad k=2,\ldots,n$$

向后代入求解UX=Y

$$x_n=y_n/u_{nn}$$

$$x_k=(y_k-\sum_{j=k+1}^{n}u_{kj}x_j)/u_{kk}$$

$$k=n-1,\ldots,1$$

计算原理如图所示。

```
function X=LU_solve(A,b)
    n=size(A,1);
    L=eye(n);
    U=A;
    %LU分解
    for k=1:n
        for i=k+1:n
            L(i,k)=U(i,k)/U(k,k);
            U(i,k:n)=U(i,k:n)-L(i,k)*U(k,k:n);
        end
```

```
    end
    %前向代入求解Ly=b
    y=zeros(n,1);
    for i=1:n
        y(i)=b(i)-L(i,1:i-1)*y(1:i-1);
    end
    %后向代入求解Ux=y
    X=zeros(n,1);
    for i=n:-1:1
        X(i)=(y(i)-U(i,i+1:n)*X(i+1:n))/U(i,i);
    end
end
```

3.3.3 迭代法

雅各比迭代法

$$x_i^{k+1}=\frac{1}{a_{ii}}(b_i-\sum_{j=1,j\neq i}^{n}a_{ij}x_j^k)$$

$$|\varepsilon_a|_i=|\frac{x_i^{k+1}-x_i^k}{x^{k+1}}|\times 100\%$$

基于以上的迭代公式以及近似相对百分比误差，代码如下

```
function [x,k]=jacobi(a,b,x0,epsilon)
    n=size(a,1);
    x=zeros(1,n);
    flag=0;%跳出迭代标志
    k=0;%迭代次数
    while 1
        k=k+1;
        for i=1:n
            sum=0;
            for j=1:n
                if i~=j
                    sum=sum+a(i,j)*x0(j);
                end
            end
            x(i)=(b(i)-sum)/a(i,i);
            if abs((x(i)-x0(i))/x(i))<epsilon%对每个都判断，即使判断出来了，也先将这一次的都迭代完，再跳出迭代
                flag=1;
            end
            x0(i)=x(i);
        end
        if flag==1
            break;
        end
        if k==1000
        break;
        end
    end
end
```

高斯-赛德尔迭代法

$$x_i^{k+1}=\frac{1}{a_{ii}}(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{k+1}-\sum_{j=i+1}^{n}a_{ij}x_j^k)$$

基于以上的迭代公式,代码如下

```
function [x,k]=GS(a,b,x0,epsilon)
    n=size(a,1);
    x=zeros(1,n);
    flag=0;
    k=0;
    while 1
    k=k+1;
    for i=1:n
        sum1=0;
        sum2=0;
        if i~=1
            for j=1:i-1
                sum1=sum1+a(i,j)*x(j);
            end
        end
        for j=i+1:n
            sum2=sum2+a(i,j)*x0(j);
        end
        x(i)=(b(i)-sum1-sum2)/a(i,i);
        if abs((x(i)-x0(i))/x(i))<epsilon
            flag=1;
        end
        x0(i)=x(i);
    end
    if flag==1
        break;
    end
    if k==1000
        break;
    end
end
end
```

3.4 算法选择

至于哪种方法最好用，挨个试一下就可以了，实验前，我对迭代法不太看好，虽然迭代法算法简单，精度优秀，但是担心能否收敛到满足全为整数的结果(即比例为有理数)。

结果：

```
    6
    2
    3

    6                2                3

    6                2                3

  138

    6                2                3

  139
```

发现四种算法都能完美完成任务。

我们再选用别的方程进行实验：

$$\mathbf{H_2 + Ca(CN)_2 + NaAlF_4 + FeSO_4 + MgSiO_3 + KI + H_3PO_4 + PbCrO_4 + BrCl + CF_2Cl_2 + SO_2 \longrightarrow PbBr_2 + CrCl_3 + MgCO_3 + KAl(OH)_4 + Fe(SCN)_3 + PI_3 + Na_2SiO_3 + CaF_2 + H_2O}$$

选用这个方程式最终得到的系数矩阵为：

```
列 1 至 17

    0     0     0     0     0     3     0     0     0     0     0     0     0    -4     0     0     0
    2     0     0     0     0     0     0     0     1     0     0     0    -1     0    -3     0     0
    2     0     0     0     0     0     0     0     0     0     0     0     0     0    -3     0     0
    1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0    -2
    0     1     0     0     0     0     0     0     0     0     0     0     0    -1     0     0     0
    0     4     0     0     0     0     0     0     2     0     0     0     0     0     0     0     0
    0     0     1     0     0     0     0     0     0     0     0     0     0     0    -1     0     0
    0     0     1     0     0     0     0     0     0     1     0     0     0     0    -3     0     0
    0     0     4     3     0     4     4     0     0     2     0     0    -3    -4     0     0    -3
    0     0     0     1     0     0     0     0     0     0     0     0    -1     0     0     0     0
    0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0    -1
    0     0     0     0     1     0     0     0     0     0     0     0     0    -1     0     0     0
    0     0     0     0     1     0     0     0     0     0     0     0     0     0     0    -3     0
    0     0     0     0     0     1     0     0     0     0     0     0     0     0     0    -1     0
    0     0     0     0     0     0     1     0     0     0    -1     0     0     0     0     0     0
    0     0     0     0     0     0     1     0     0     0     0    -1     0     0     0     0     0
    0     0     0     0     0     0     0     1     0     0    -2     0     0     0     0     0     0
    0     0     0     0     0     0     0     1     2     0     0    -3     0     0     0     0     0

列 18 至 19

    0    -2
    0     0
    0     0
   -1     0
    0     0
    0     0
   -2     0
    0     0
    0     0
    0    -1
    0     0
    0     0
    0     0
    0     0
    0     0
    0     0
    0     0
    0     0
    0     0
```

常数项为

```
列 1 至 17

  -2     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

列 18 至 19

    0     0
```

但当我们解方程时，发现无论用高斯消元法，LU分解法，迭代法，得到的结果都是清一色的NaN

```
列 1 至 17

   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

列 18 至 19

   NaN   NaN
```

但是用matlab自带的函数A\b'求解却可以解出来

```
0.1705
0.0682
0.1136
0.0341
0.0682
0.0227
0.0682
0.1364
0.0341
0.2273
0.0682
0.0682
0.0341
0.0682
0.1136
0.0227
0.0341
0.1705
0.8977
```

此为用列主元消去法也可以得到相同的结果：

```
列 1 至 10

  0.1705    0.0682    0.1136    0.0341    0.0682    0.0227    0.0682    0.1364    0.0341    0.2273

列 11 至 19

  0.0682    0.0682    0.0341    0.0682    0.1136    0.0227    0.0341    0.1705    0.8977
```

理论上A的秩为19，常数项不为零，未知量也为19，应该存在唯一解。

为了保证其他算法没有问题，我用之前的作业题验证一下：

$$\begin{bmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 0.0014 \\ 0.0022 \\ 0.0025 \\ 0.0022 \\ 0.0014 \end{bmatrix}$$

```
[X,h]=separate(7,0,120);
n=length(X);%点数
A=zeros(n-2,n);%定义初始0矩阵
B=[];%常数向量
for i=2:n-1
    k=5*X(i)-X(i)*X(i)/24;
    k=k*h*h/24000000;%常数项
    B=[B,k];
    A(i-1,i-1)=1;
    A(i-1,i)=-2;
    A(i-1,i+1)=1;
```

```
end
A_re=A(:, 2:n-1);
X_re=X(2:n-1);
disp(A_re);
disp(B);
x0=[1,1,1,1,1];
disp(Gauss(A_re,B));
disp(LU_solve(A_re,B));
disp(jacobi(A_re,B,x0,eps));
disp(GS(A_re,B,x0,eps));
```

上述为第三章作业中获得矩阵与常数项的代码，在此处用四种算法，均能得到正确的结果。

```
-0.0049    -0.0083    -0.0096    -0.0083    -0.0049

-0.0049
-0.0083
-0.0096
-0.0083
-0.0049

-0.0049    -0.0083    -0.0096    -0.0083    -0.0049

-0.0049    -0.0083    -0.0096    -0.0083    -0.0049
```

这说明，我们得到的化学方程组矩阵因为有太多的0可能是病态或者说更恶劣的，以至于无法求解。但是列主元消去法，通过调换主元的方法可以得到正确答案。
我们再回看原方程组得到的答案，对所有数乘88之后，得到如下结果：

```
列 1 至 10

 15.0000    6.0000   10.0000    3.0000    6.0000    2.0000    6.0000   12.0000    3.0000   20.0000

列 11 至 19

  6.0000    6.0000    3.0000    6.0000   10.0000    2.0000    3.0000   15.0000   79.0000
```

恰好与本题答案相同，所以用解线性方程组的方法来配平方程式是可行的，只是在解方程式的算法中，有些算法可能会因为遇到病态方程组或者除数为0的情况等等而无效，列主元消去法就不会这样。

$$
\begin{aligned}
&88\mathrm{H_2} + 15\mathrm{Ca(CN)_2} + 6\mathrm{NaAlF_4} + 10\mathrm{FeSO_4} + 3\mathrm{MgSiO_3} \\
&+ 6\mathrm{KI} + 2\mathrm{H_3PO_4} + 6\mathrm{PbCrO_4} + 12\mathrm{BrCl} + 3\mathrm{CF_2Cl_2} \\
&+ 20\mathrm{SO_2} \\
&\longrightarrow \\
&6\mathrm{PbBr_2} + 6\mathrm{CrCl_3} + 3\mathrm{MgCO_3} + 6\mathrm{KAl(OH)_4} + 10\mathrm{Fe(SCN)_3} \\
&+ 2\mathrm{PI_3} + 3\mathrm{Na_2SiO_3} + 15\mathrm{CaF_2} + 79\mathrm{H_2O}
\end{aligned}
$$

因此我们选择列主元消去法。

3.5 后续处理

我们得到的解一般为小数，但是其肯定都是有理数，那么一定可以化为全为整数的情况。

```
列 1 至 7

    15/88          3/44          5/44          3/88          3/44          1/44          3/44

列 8 至 14

     3/22          3/88          5/22          3/44          3/44          3/88          3/44

列 15 至 19

     5/44          1/44          3/88         15/88         79/88
```

由图很容易可以看出应该乘88。

因此我们可以先format rat输出改为分数，然后把每个分数取出来，求他们的最小公倍数，然后全部乘最小公倍数即可。

```
format rat;
X=ColMajorGauss(A_temp,b);
disp(X(1));
D_array=[];
for i=1:length(X)
    [I,D]=numden(sym(X(i)));
    D=eval(D);
    D_array=[D_array,D];
end
disp(D_array);
tool=D_array(1);
for i=2:length(X)
    if isinteger(tool/D_array(i))
        tool=tool*gcd(tool,D_array(i));
    end
end
```

然后再美化一下输出

```
X=X*tool;
disp(X);
disp("最小公倍数为")
disp(tool);
fprintf("%d",tool);
fprintf(cheminput(1));
for i=2:length(cheminput)
    fprintf("+");
    fprintf("%d",int32(X(i-1)));
    fprintf(cheminput(i));
end
fprintf("=\n");
fprintf("%d",int32(X(length(cheminput))));
fprintf(chemoutput(1));
for i=2:length(chemoutput)
    fprintf("+");
    fprintf("%d",int32(X(length(cheminput)+i-1)));
    fprintf(chemoutput(i));
```

```
end
```

就可以直接输出化学方程式

```
88H2+6Ca(CN)2+10NaAlF4+3FeSO4+6MgSiO3+2KI+6H3PO4+12PbCrO4+3BrCl+20CF2Cl2+6SO2=
6PbBr2+6CrCl3+3MgCO3+6KAl(OH)4+10Fe(SCN)3+2PI3+3Na2SiO3+15CaF2+79H2O>>
```

3.6 其他测试

接下来测试其他的方程式

$$10\,K_4Fe(CN)_6 + 122\,KMnO_4 + 188\,H_2SO_4 == 60\,CO_2 + 60\,KNO_3 + 188\,H_2O + 51\,K_2SO_4 + 122\,MnSO_4 + 5\,Fe_2(SO_4)_3$$

输出为：

```
10K4Fe(CN)6+122KMnO4+188H2SO4=
60CO2+60KNO3+188H2O+51K2SO4+122MnSO4+5Fe2(SO4)3>>
```

结果相同

6 K4[Fe(SCN)6⁺] + 97 K2Cr2O7 + 355 H2SO4 → 3 Fe2(SO4)3 + 97 Cr2(SO4)3 + 36 CO2 + 355 H2O + 91 K2SO4 + 36 KNO3

```
6K4Fe(SCN)6+97K2Cr2O7+355H2SO4=
3Fe2(SO4)3+97Cr2(SO4)3+36CO2+355H2O+91K2SO4+36KNO3>>
```

4 可改进之处

首先是方程组的问题，矩阵的特殊性使得高斯消元法、LU分解法和迭代法的不稳定，只有列主元消元法才能正常工作。怎样构造矩阵，使得其他方法可以正常工作，可以提高算法的速度。

其次在构造矩阵的时候，为了构造方阵进行了先消行，再消列的操作，消行一般没有问题，消列一般有问题。对于大部分的方程式，最后得到的结果为矩阵的秩比未知量个数小1，即只需要赋一个值即可满足；但是存在一部分方程式，最终秩的比未知量个数小2甚至更多，这就导致了两个问题：消哪两列和怎么赋值。对于前者，我们可以用和消行同样的方法，随机数消去，然后看剩余矩阵的秩是否为原矩阵的秩；但是后者怎么赋值有些难，虽然方程组的配平有无数组解，但是其之间的比例是不变的。这个赋值就相当于要先赋值一个比例进去，然后再进行运算，涉及符号运算，有些困难。在实验如下方程组时就遇到了该问题：

$$32\,Si_5C_{18}H_{45}ClBr_2 + 96\,KC_8 == 7\,Si_{21}C_{72}H_{180} + 12\,SiC_4H_{12} + SiC_{16}H_{36} + 32\,KCl + 64\,KBr + 776\,C$$

得到的原矩阵是6×8的矩阵，一开始消前两列，得到的秩为5，然后改变算法，随机消去，秩满足要求了，但是赋值给了两个1，导致最终的结果出现了负数，才意识到这个问题。理论上修改秩问题的代码如下。

```
A_temp_temp=A_temp;
A_temp_temp(:,1:give_num)=[];
while rank(A_temp_temp)~=rank(A)
    A_temp_temp=A_temp;
    randomint=randperm(b,give_num);
    B=[];
    for i=1:rank(A)
        sum=0;
```

```
        for j=1:give_num
            sum=sum+A_temp(i,randomint(j));
        end
        B=[B,-sum];
    end
    A_temp_temp(:,randomint)=[];
end
```

再者，本程序也有一定的局限性。实际化学过程中，存在好几个方程式线性组合的方程式，这种单看组合的方程式，可能不止一个解；本程序只写了只有一层括号的方程式的分解，实际上肯定有很多两层以上的化学式，他的实现也不难，先按照最外层括号分成若干块，然后对每块都用先前定义的unpack_complex_ele函数进行递归，直到没有括号。另外，还有一种特殊的反应——聚合反应，其生成物下标为n，反应物系数也有n，具体实现应该涉及带符号计算的解线性方程组。

5 个人感想

通过本次大作业，我对高斯消元法，LU分解以及迭代法等用于解线性方程组的算法的使用条件有了更深刻的认知，对于一些奇异或者病态的方程组，这些算法可能得出错误解，甚至会无效。但是列主元消元法稳定性非常好，因为其实时更新主元的特性。

很多现实问题的本质上都是数值计算问题，关键和难点并不在于求解，而是数学建模过程，以本题为例，解线性方程组的算法都在课上讲过，也在作业中实现过，难的地方主要在将化学式分解为元素，并且建立合适的方阵和选取适合的算法，以及无尽的debug过程。