プログラム実習（Oクラス）

第3回：課題「行列計算」の説明

行列計算

画像処理、自然現象のシミュレーション

例）静電気のポワソン方程式（プラス境界条件）

$$\frac{\partial^2 V(x)}{\partial x^2} = -\frac{\rho(x)}{\varepsilon_0} \quad \rho(x): \text{given}$$

数値シミュレーションでは差分方程式
（格子点$x=i\Delta x$、$i=0,1,2,\ldots..$）

$$V_{i-1} - 2V_i + V_{i+1} = -\frac{(\Delta x)^2}{\varepsilon_0}\rho_i$$

行列形式にまとめられ、

$$\begin{bmatrix} \ddots & & & & & & \\ & 1 & -2 & 1 & & 0 & \\ & & 1 & -2 & 1 & & \\ & 0 & & 1 & -2 & 1 & \\ & & & & & & \ddots \end{bmatrix} \begin{pmatrix} \vdots \\ V_{i-1} \\ V_i \\ V_{i+1} \\ \vdots \end{pmatrix} = \left(-\frac{(\Delta x)^2}{\varepsilon_0} \right) \begin{pmatrix} \vdots \\ \rho_{i-1} \\ \rho_i \\ \rho_{i+1} \\ \vdots \end{pmatrix}$$

逆行列、連立一次方程式の解法に帰着

$$\begin{pmatrix} \vdots \\ V_{i-1} \\ V_i \\ V_{i+1} \\ \vdots \end{pmatrix} = \left(-\frac{(\Delta x)^2}{\varepsilon_0} \right) \begin{bmatrix} \ddots & & & & & & \\ & 1 & -2 & 1 & & 0 & \\ & & 1 & -2 & 1 & & \\ & 0 & & 1 & -2 & 1 & \\ & & & & & & \ddots \end{bmatrix}^{-1} \begin{pmatrix} \vdots \\ \rho_{i-1} \\ \rho_i \\ \rho_{i+1} \\ \vdots \end{pmatrix}$$

本課題

連立一次方程式の数値解法

１）実数を行列とする行列クラスRMatrix

２）複素数を行列とする行列クラスCMatrix

３）Gaussの消去法（連立一次方程式解法のアルゴリズム）のRMatrixクラスへの実装

＝＝＝＝＝

発展）Gaussの消去法のCMatrixクラスへの実装

ファイルのコピー

javaのサンプルプログラムが与えられている。

1)TestMax22.java,2)Rmatrix.java,3)TestRMatrix01.java,4)Complex.java,5)TestComplex01.java,6)TestGauss01.java

これらに追加、発展を行い課題を遂行する。

```
$ cp ~yamasita/prac1/*.* .
```

実数行列RMatrixの実装

NxM行列の和、差、積を返す行列クラスの実装

課題1）まず2x2行列で慣れる

TestMax22.java中の行列クラスMax22の完成。

```
class Max22 { // 2x2 配列クラス(配列要素は double 型とする)

  private double[][] m; // フィールド変数として配列参照変数を宣言

  //コンストラクタ (配列の実体を生成し，初期値を代入する)
  Max22( double[][] a ){
    //引数として受け取った配列の中身を，新たに生成した配列に代入する
    m = new double[2][2];
    m[0][0]=a[0][0];
    m[0][1]=a[0][1];
    m[1][0]=a[1][0];
    m[1][1]=a[1][1];
  }
   // 行列の和を計算する．引数として 2x2 行列である Max22 オブジェクトを
   // 受け取り(max)，加算した結果の行列の参照を戻り値として返す
   Max22 add( Max22 max ) {
     //加算結果を格納する行列 ans を生成し，まず自身の値を代入
     Max22 ans = new Max22( m );
     //ans に，引数の行列 max を加算する
     ans.m[0][0] += max.m[0][0]; ans.m[0][1] += max.m[0][1];
     ans.m[1][0] += max.m[1][0]; ans.m[1][1] += max.m[1][1];
     return ans ;
   }
   //差演算 subtract()、および積演算 multiply()を実行するメソッドを挿入せよ

   // 行列要素を出力する
  void printMatrix() {
    System.out.print( "a00=" + m[0][0] );
    System.out.println( " a01=" + m[0][1] );
    System.out.print( "a10=" + m[1][0] );
    System.out.println( " a11=" + m[1][1] );
  }
}
```

課題2） 課題1を参考にして一般のNxM行列の和、差、積を計算するクラスRMatrix（ファイルRmatrix.java中にある）を完成。

注）行列をNxMに一般化したので、和、差の演算では演算する行列同士の行数と列数がそれぞれ一致しているか判定する。積ABにおいては、Aの列数とBの行数が一致していることを判定する。

RMatrix.javaの一部

```
//コンストラクタ２ (row行 column列の配列オブジェクトを生成し，０で初期化する)
RMatrix( int row, int column ){
  m = new double[row][column];
  for (int i=0; i < row; i++ ) {
    for(int j=0; j < column; j++ ) {
       m[i][j] = 0.0;
    }
  }
}
// 行列の和  m1.add(m2) = m1 + m2
RMatrix add( RMatrix max ) { //加算した行列を戻り値として返すメソッド
  // 行数と列数が一致しているかどうかを調べる
  if( m.length != max.m.length || m[0].length != max.m[0].length ) {
     System.out.println( "Matrix addition not defined!" );
     return null;
  }
  //結果を格納する行列 ans を生成し，まず自身の値を代入
  RMatrix ans = new RMatrix( m );
  for (int i=0; i < m.length; i++ ) {
    for(int j=0; j < m[0].length; j++ ) {
       ans.m[i][j] += max.m[i][j];
    }
  }
  return ans ;
}
// 行列の差 m1.subtract(m2) = m1 - m2 当然だが演算の順番によって答えが変わる！
// 実装せよ

// 行列の積 m1.mutiply(m2) = m1 * m2  演算の順番に注意！
RMatrix multiply( RMatrix max ) { //積算した行列を戻り値として返すメソッド
  // 積を実行できるかどうかを調べる．AB のとき，A の列数と B の行数が一致しなければならない
  if( m[0].length != max.m.length ) {
     System.out.println( "Matrix multiplication not defined!" );
     return null;  // 例外処理を行うべきであるが本課題では省略
  }
  //N×M 行列と M×K 行列の積は N×K 行列になる．
  //まず結果を格納する行列 ans を生成し，０で初期化


  // この部分のコードを埋めなさい．

}
// 行列要素を出力するメソッド
void printMatrix() {
```

複素行列CMatrixの実装

複素数を要素として持つNxM行列の和、差、積を返す行列クラスの実装

課題3）まず複素数に対して和、差、積、商を計算できるようにする。Complex.java中のクラスComplexを完成せよ。

Complex.java

```
class Complex { //複素数クラス
  private double re;  // 実部
  private double im;  // 虚部

  public Complex (double x, double y) { //コンストラクタ(実部,虚部)
    re = x; // 実部
    im = y; // 虚部
  }
  public double re() { // 実部を返すメソッド
    return re;
  }
  public double im() { // 虚部を返すメソッド
    return im;
  }
  //複素数の大きさ|z|を返すメソッド
  public double abs() {
    return  Math.sqrt( re * re + im * im );
  }
  //複素数の和を返すメソッド
  public Complex add ( Complex c ) {
    return  new Complex( re + c.re, im + c.im );
  }
  //複素数の差を返すメソッドsubtract()を実装せよ z1.subtract(z2) = z1 - z2
  //複素数の積を返すメソッドmultiply()を実装せよ z1.multiply(z2) = z1 * z2
  //複素数の商を返すメソッドdivide()を実装せよ z1.divide(z2) = z1 / z2
```

課題4）RMatrixクラスとComplexクラスを用いて、一般のNxM複素行列の和、差、積を計算できるようにするCMatrixクラスを完成させよ。

・Complexクラスは変更せずそのまま仕様し、Rmatirxの扱える変数範囲を複素数に拡張する。

・テキスト中に示された仕様にのっとること。

（フィールド、コンストラクタ、メソッド）

本テーマ最終課題

連立一次方程式を解くメソッドを
行列クラスRMatrixへ加える

仕様アルゴリズム：Gaussの消去法

例題）

$$\begin{cases} 2x & -3y & +z & = & 1 & (1) \\ x & +2y & -3z & = & 4 & (2) \\ 3x & +2y & -z & = & 5 & (3) \end{cases}$$

を解く

（１）と（２）を入れ替える解く

$$\begin{cases} x & +2y & -3z & = & 4 & (2) \\ 2x & -3y & +z & = & 1 & (1) \\ 3x & +2y & -z & = & 5 & (3) \end{cases}$$

$$\begin{cases} x & +2y & -3z & = & 4 & (2) \\ 2x & -3y & +z & = & 1 & (1) \\ 3x & +2y & -z & = & 5 & (3) \end{cases}$$

（2）の−2倍を（1）へ、−3倍を（3）へ加える

$$\begin{cases} x & +2y & -3z & = & 4 & (1') \\ & -7y & +7z & = & -7 & (2') \\ & -4y & +8z & = & -7 & (3') \end{cases}$$

xの消去に用いた1行1列目の要素（今は1だった）をピボットと呼びます。

$$\begin{cases} x & +2y & -3z & = & 4 & (1') \\ & -7y & +7z & = & -7 & (2') \\ & -4y & +8z & = & -7 & (3') \end{cases}$$

（２‘）を－1/7倍する。

$$\begin{cases} x & +2y & -3z & = & 4 & (1') \\ & y & -z & = & 1 & (2'') \\ & -4y & +8z & = & -7 & (3') \end{cases}$$

$$\begin{cases} x & +2y & -3z & = & 4 & (1') \\ & y & -z & = & 1 & (2'') \\ & -4y & +8z & = & -7 & (3') \end{cases}$$

（2“）を4倍して（3‘）へ加える。

$$\begin{cases} x & +2y & -3z & = & 4 & (1') \\ & y & -z & = & 1 & (2'') \\ & & 4z & = & -3 & (3'') \end{cases}$$

$$\begin{cases} x & +2y & -3z & = & 4 & (1') \\ & y & -z & = & 1 & (2'') \\ & & 4z & = & -3 & (3'') \end{cases}$$

$(3')$より、

$$z = -\frac{3}{4}$$

これを$(2'')$へ代入して、

$$y = \frac{1}{4}$$

以上を$(1')$へ代入して、

$$x = \frac{5}{4}$$

が得られた。

連立一次方程式へ行った操作、

（１）方程式の順番を並びかえる

（２）一つの方程式にある数を掛け算して方程式へ加える。

（３）一つの方程式に0でない数を掛け算する。

こうして簡単な連立一次方程式へ変形して行く。

ガウスの消去法

行列による一般表現

n変数の連立一次方程式

$$
\begin{aligned}
a_{11}x_1 \quad &+a_{12}x_2 \quad +\cdots \quad +a_{1n}x_n = b_1 \\
a_{21}x_1 \quad &+a_{22}x_2 \quad +\cdots \quad +a_{2n}x_n = b_2 \\
\vdots \quad &\quad \vdots \quad \quad \vdots \quad \quad \vdots \\
a_{n1}x_1 \quad &+a_{n2}x_2 \quad +\cdots \quad +a_{nn}x_n = b_n
\end{aligned}
$$

⇩ 行列で表現

$$\mathbf{A}\vec{x} = \vec{b}$$

ここで、

$$\mathbf{A}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad \vec{x}=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \vec{b}=\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

A行列とbベクトルをまとめて

$$n\times(n+1)\text{拡大係数行列}:[A:b]=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix}$$

$$\tilde{A}^{(1)} = [\tilde{A} : b] = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & a_{2,n+1}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{pmatrix}$$

ここで，$a_{11}^{(1)} \neq 0$ ならば，1 行目に対して $\left(\frac{1}{a_{11}^{(1)}} R_1 \to R_1\right)$ を行って，先頭の行列要素を 1 とする．

$$\tilde{A}^{(1)\prime} = \begin{pmatrix} 1 & a_{12}^{(1)\prime} & \cdots & a_{1n}^{(1)\prime} & a_{1,n+1}^{(1)\prime} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & a_{2,n+1}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{pmatrix}$$

R_1は拡大係数行列の1行目の行ベクトル

$$\tilde{A}^{(1)\prime}=\begin{pmatrix} 1 & a_{12}^{(1)\prime} & \cdots & a_{1n}^{(1)\prime} & a_{1,n+1}^{(1)\prime} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & a_{2,n+1}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{pmatrix}$$

2行目以下の行jに対して　$(-a_{j1}^{(1)}R_1+R_j \rightarrow R_j)$を行って，係数$a_{j1}^{(1)}$($j=2\sim$n)を消去する．

$$\tilde{A}^{(2)}=\begin{pmatrix} 1 & a_{12}^{(1)\prime} & \cdots & a_{1n}^{(1)\prime} & a_{1,n+1}^{(1)\prime} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_{n,n+1}^{(2)} \end{pmatrix}$$

$$\tilde{A}^{(2)} = \begin{pmatrix} 1 & a_{12}^{(1)\prime} & \cdots & a_{1n}^{(1)\prime} & a_{1,n+1}^{(1)\prime} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_{n,n+1}^{(2)} \end{pmatrix}$$

次に，$a_{22}^{(2)} \neq 0$ ならば，2 行目に対して $\left(\frac{1}{a_{22}^{(2)}} R_2 \to R_2\right)$ を行って，先頭の行列要素を 1 とする．

$$\tilde{A}^{(2)\prime} = \begin{pmatrix} 1 & a_{12}^{(1)\prime} & \cdots & a_{1n}^{(1)\prime} & a_{1,n+1}^{(1)\prime} \\ 0 & 1 & \cdots & a_{2n}^{(2)\prime} & a_{2,n+1}^{(2)\prime} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_{n,n+1}^{(2)} \end{pmatrix}$$

$$\tilde{A}^{(2)'}=\begin{pmatrix} 1 & a_{12}^{(1)'} & \cdots & a_{1n}^{(1)'} & a_{1,n+1}^{(1)'} \\ 0 & 1 & \cdots & a_{2n}^{(2)'} & a_{2,n+1}^{(2)'} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_{n,n+1}^{(2)} \end{pmatrix}$$

3行目以下の行 j に対して $(-a_{j2}^{(2)}R_2+R_j \rightarrow R_j)$ を行って，係数 $a_{j2}^{(2)}$ $(j=3\sim\mathrm{n})$を消去する．

$$\tilde{A}^{(3)}=\begin{pmatrix} 1 & a_{12}^{(1)'} & a_{13}^{(1)'} & \cdots & a_{1n}^{(1)'} & a_{1,n+1}^{(1)'} \\ 0 & 1 & a_{23}^{(2)'} & \cdots & a_{2n}^{(2)'} & a_{2,n+1}^{(2)'} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & a_{3,n+1}^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & a_{n,n+1}^{(3)} \end{pmatrix}$$

最終的に、係数拡大行列

$$\tilde{A}^{(n)'}=\begin{pmatrix} 1 & a_{12}^{(1)'} & a_{13}^{(1)'} & \cdots & a_{1,n-1}^{(n-1)'} & a_{1n}^{(1)'} & a_{1,n+1}^{(1)'} \\ 0 & 1 & a_{23}^{(2)'} & \cdots & a_{2,n-1}^{(n-1)'} & a_{2n}^{(2)'} & a_{2,n+1}^{(2)'} \\ 0 & 0 & 1 & \cdots & a_{3,n-1}^{(n-1)'} & a_{3n}^{(3)'} & a_{3,n+1}^{(3)'} \\ 0 & 0 & 0 & \cdots & \vdots & \vdots & \\ \vdots & \vdots & \vdots & \cdots & 1 & a_{n-1,n}^{(n-1)'} & a_{n-1,n+1}^{(n)'} \\ 0 & 0 & 0 & \cdots & 0 & 1 & a_{n,n+1}^{(n)'} \end{pmatrix}$$

が得られる。

重要！

対角要素a_{ii}で割り算を行っている。

$a_{ii}=0$なら零割で実効エラー

・p行i列、ただしi<p、でa_{pi}が0でない行を見つけ、i行とp行を入れ替える。

・ p行i列、ただしi<p、でa_{pi}が最大となる行を見つけ、i行とp行を入れ替える。（ピボット選択：a_{ii} が0でなくても非常に小さな数の場合はオーバーフローとなる。）

ガウスの消去法のRMatrixへの実装

課題５）

メソッド：

solveSimultaneousLinearEquations()

をRMatrixクラスへ組み込む

```
public double []
solveSimultaneousLinearEquations(doubl
e[] rightHandValues)
```

はAx=bのbを引数として与えられ、xを返す。

```
double[] solveSimultaneousLinearEquations( double[] b ){

      //this の行列が正方行列であるか，引数の配列が正方行列の
      //サイズと合致するかチェックを行う
      if( m.length != m[0].length || b.length != m.length )
          System.out.println("正方行列ではないので解けません．");
          return null;
      }
      int n = m.length; // 正方行列の一辺の長さを与える
      // 拡大係数行列のコンストラクト
      RMatrix Aex = new RMatrix( n, n + 1 );
      // 拡大係数行列の係数行列Aの部分のコピー
      for (int i=0;i<n;i++){
        for (int j=0;j<n;j++) {
          Aex.m[i][j] = m[i][j];
        }
      }
      // 拡大係数行列の定数ベクトルBの部分のコピー
      for(int i=0;i<n;i++) Aex.m[i][n] = b[i];
      // 解を格納する配列を生成＆初期化
      double[] ans = new double[n];
      // 作業用バッファ
      double[] work = new double[n+1];

//デバッグ用拡大係数行列の出力(完成したら下記２行はコメントアウトする
      Aex.printMatrix();
      System.out.println();

      //対角要素が０の場合は０でない行と入れ替える．０でない，というだ
      //なるべく絶対値の大きい要素を選んだ方が，割り算の際の桁あふれの
      //同時に減らせる．
      int n_pivot = 0; // ピボットの配列添え字
      double max;      // 最大値を格納する変数
      double pivot;    // ピボットの数値を格納する変数
      double q;        // 作業用

      // 上から順に各行について調べていくが，最終行については調べる必
      // i は行の配列添え字
      for( int i = 0; i < n-1; i++){
        max = 0.0;
        n_pivot = i;
        //対角要素の絶対値が一番大きい係数を求める
        for(int j = i; j < n; j++){
          if( Math.abs( Aex.m[j][i] ) > max ) {
            n_pivot = j;
            max = Math.abs( Aex.m[j][i] );
          }
        }
        // 最大の係数が０にとても近い（逆数を取ったらオーバーフローする）場合
        if ( Math.abs(max) < 1.0/Double.MAX_VALUE ){
          System.out.println("０でないピボットが見つからない！");
          return null; //終了
        }
        // 対角要素が最大でない場合は行を入れ替える
        if ( n_pivot != i ){
            for(int k=0;k<n+1;k++){
              work[k]=Aex.m[n_pivot][k];
              Aex.m[n_pivot][k]=Aex.m[i][k];
              Aex.m[i][k]=work[k];
            }
        }
      } // 行の入れ替え完了（次ページへ続く→）
```

```
      //対角要素から下半分を消去する

      // この部分を埋めよ


      // 解を求める
      for(int i=n-1;i>=0;i--) {

      // この部分を埋めよ

      }

//デバッグ用拡大係数行列の出力(完成したらコメントアウトする)
      Aex.printMatrix();
      System.out.println();

      return ans;
}
// メソッドの記述終了
```

空欄を埋め、RMatrixへ組み込む

プログラムの動作チェック

クラス

Max22,RMatrix,Complex

は、それぞれ2x2,NxMの実数行列、複素数の和までそのままで実効できるようになっている。

それぞれ実効用javaファイル

TestMax22.java,TestRMatrix01.java,TestComplex01.java

を用いて動作を確認せよ。

★これが終われば帰ってよし！

次回

１０月１６日午前１３:２０～

「内容」行列計算課題の事前レポート作成

注）家でやってきても可。その場合はTAのチェックが通ったらその場で帰ってよし。

```
$ mkdir prac1
$ cd prac1
$ cp ~yamasita/prac1/*.* .
```