ＪＡＶＡによる有限要素法の基礎的研究

永　藤　壽　宮*

A Fundamental Study of Finite Element Methods In JAVA

NAGATO Toshimiya*

JAVA is now considered the most powerful and complete programming language, giving possibility to graphic interface, advanced calculations, etc. So I found that the use of such a tool should give big opportunities and possibilities to programming in the civil engineering field.

Program of finite element methods in JAVA is investigated for two purposes.
First purpose: To use JAVA programming for civil engineering problems.
Second purpose: To compare JAVA-runtime with Fortran-runtime.

キーワード：JAVA, Fortran, 有限要素法

1．はじめに

本研究の目的は，筆者がこれまでに開発した有限要素法プログラムを(2 次元弾性問題から有限変位解析まで)，JAVA を使って再プログラム化し，Fortran と JAVA について比較し，建設の分野での有効性について考察することを目的とした．

JAVA は，Sun Microsystems 社が開発したプログラミング言語で，C言語に似た表記法を採用しているが，C言語など既存の言語の欠点を踏まえて一から設計された言語である．今までの言語にない完全なオブジェクト指向性を備えているとされている．

また，強力なセキュリティ機構や豊富なネットワーク関連の機能が標準で搭載されており，ネットワーク環境で利用されることを強く意識した仕様になっている．

JAVA では約 80 パッケージを標準で含んでいる．
例えば，以下のような
java.math：意精度の整数演算，および任意精度の 10 進数演算を実行するためのクラスを提供する．
java.awt：ユーザインタフェースの作成およびグラフィックスとイメージのペイント用のすべてのクラスを含む．
java.io：データストリーム，直列化，およびファイルシステムによるシステム入出力用に提供されている．
などがあげられる．そしてソフトから COMPILER まで全て Sun Microsystems の HP で提供されている．

2．有限要素法の概要

解析しようとする構造物を，有限個の簡単な形状をした要素の集合体として考える有限要素法の利用によって構造物全体の詳細な解析を可能とし，構造物の断面力やたわみなどを計算できることは，周知のことである．

まず 2 次元弾性問題のプログラムから始まり，有限変位解析まで JAVA コードで書き換えた．その際，最近開発された行列計算用パッケージ JAMA を利用し，マトリックスの足し算掛け算や逆マトリックスなどを一部利用した．

例として以下のように

```
public Matrix plus(Matrix B)
        Matrix C = Matrix A + B;
public Matrix inverse(Matrix B)
        Matrix A = B.inverse();
```

などを挙げておく．JAVA のクラスライブラリは，http://math.nist.gov/javanumerics/などを参照して加えていった．

浮動小数点の問題がどのように Fortran 演算時間に影響するかを考察する．

* 長野工業高等専門学校環境都市工学科助教授
原稿受付　2004 年 5 月 20 日

3. JAVA CODE プログラムの一部

下記に2次元弾性問題のプログラムの一部を掲載する.

```
/*
 * main_entry.java
 *
 * Created on November 1, 2003, 9:29 PM
 */
package DynamicInputData1;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import Jama.*;
import javax.swing.*;
import javax.swing.ImageIcon;
import javax.swing.JTabbedPane;
import javax.swing.text.*;
import javax.swing.event.*;
import javax.swing.table.*;
/**
 * @author nagato
 */
/** Initializes the Form */
public class main_entry extends
javax.swing.JFrame {

    public main_entry() {
        initComponents();
        pack();
    }
    // Variables declaration
    // public javax.swing.JFrame
jMainFrame;
    public javax.swing.JTabbedPane
tabbedPane;
    public javax.swing.JPanel MAINPanel;
    public javax.swing.JPanel jPanel1;
    public javax.swing.JPanel jPanel2;
    private javax.swing.ImageIcon icon;
    public javax.swing.JScrollPane
KAKOMscrollPane;
    public javax.swing.JTable jKAKOMTable;
    public
javax.swing.table.DefaultTableModel
DefaultKAKOMModel;
    public javax.swing.JScrollPane
POINTSscrollPane;
    public javax.swing.JTable jPOINTSTable;
    public
javax.swing.table.DefaultTableModel
DefaultPOINTSModel;
    public javax.swing.JScrollPane
KOXscrollPane;
    public javax.swing.JTable jKOXTable;
    public
javax.swing.table.DefaultTableModel
DefaultKOXModel;
    public javax.swing.JScrollPane
KOYscrollPane;
    public javax.swing.JTable jKOYTable;
    public
javax.swing.table.DefaultTableModel
DefaultKOYModel;
    public javax.swing.JScrollPane
NFscrollPane;
    public javax.swing.JTable jNFTable;
    public
javax.swing.table.DefaultTableModel
DefaultNFModel;
    public int
NSTRES,NELT,NODT,NT,KOX,KOY,NF,i,j,k,l,
m,n,MM,N;
    public double E,PO,TH,W;
    public int
KAKOM[][],NOKX[],NOKY[],NO[],INDEX[];
    public double EI[][],T[], X[],Y[],F[],FX[],
FY[],B[][],D[][],STR[][],SM[][],TSM[][],TSM1[][],
TSM2[][],DIS[],REAC[];

    private javax.swing.JLabel jlblCOMPUTE;
    private javax.swing.JLabel jlblFIRST;
    private javax.swing.JLabel jlblFIRST2;
    private javax.swing.JLabel jlblNSTRES;
    private javax.swing.JLabel jlblNb1;
    private javax.swing.JLabel jlblNb2;
    private javax.swing.JLabel jlblE;
    private javax.swing.JLabel jlblPO;
    private javax.swing.JLabel jlblKOX;
    private javax.swing.JLabel jlblKOY;
    private javax.swing.JLabel jlblNF;
    private javax.swing.JLabel jlblSum;
    private javax.swing.JTextField jtfNSTRES;
    private javax.swing.JTextField jtfNb1;
```

```
    private javax.swing.JTextField jtfNb2;
private javax.swing.JTextField jtfE;
    private javax.swing.JTextField jtfPO;
    private javax.swing.JTextField jtfKOX;
    private javax.swing.JTextField jtfKOY;
    private javax.swing.JTextField jtfNF;
    private javax.swing.JButton jbtnOK;
    private javax.swing.JCheckBox
jCheckNSTRES;
    //Graphic initialization + event handling
    private void initComponents() {
        // jMainFrame = new
javax.swing.JFrame(); // Global panel
        tabbedPane = new JTabbedPane();
        MAINPanel = new
javax.swing.JPanel(); // Global panel
        jPanel1 = new javax.swing.JPanel(); //
Global panel
        jPanel2 = new javax.swing.JPanel(); //
Global panel
        jlblCOMPUTE = new
javax.swing.JLabel(); // Label for FIRST
        jlblFIRST = new javax.swing.JLabel();
// Label for FIRST
        jlblFIRST2 = new javax.swing.JLabel();
// Label for FIRST2
        jlblNSTRES = new
javax.swing.JLabel(); // Label for NSTRES
        jlblNb1 = new javax.swing.JLabel(); //
Label for Nb1
        jlblNb2 = new javax.swing.JLabel(); //
Label for Nb2
        jlblE = new javax.swing.JLabel(); //
Label for E
        jlblPO = new javax.swing.JLabel(); //
Label for PO
        jlblKOX = new javax.swing.JLabel(); //
Label for KOX
        jlblKOY = new javax.swing.JLabel(); //
Label for KOY
        jlblNF = new javax.swing.JLabel(); //
Label for NF
        jlblSum = new javax.swing.JLabel();
//Sum
        jtfNSTRES = new
javax.swing.JTextField(); //NSTRES entry
        jtfNb1 = new javax.swing.JTextField();
//Nb1 entry
        jtfNb2 = new javax.swing.JTextField();
//Nb2 entry
        jtfE = new javax.swing.JTextField(); //E
entryjbtnOK = new javax.swing.JButton();
//Compute button
        jtfPO = new javax.swing.JTextField();
//PO entry
        jtfKOX = new javax.swing.JTextField();
//KOX entry
        jtfKOY = new javax.swing.JTextField();
//KOY entry
        jtfNF = new javax.swing.JTextField();
//NF entry
        jbtnOK = new javax.swing.JButton();
//Compute button
        jCheckNSTRES = new
javax.swing.JCheckBox();
        //Panel
        jPanel1.setLayout(new
java.awt.GridLayout(10, 10));
        jPanel2.setLayout(new
java.awt.GridLayout(10, 10));
        //
jPanel1.setBackground(java.awt.Color.BLUE);
        //FIRST label

jlblFIRST.setVerticalAlignment(JLabel.CENTE
R);

jlblFIRST.setHorizontalAlignment(JLabel.RIGH
T);
        jlblFIRST.setSize(1,0);
        jlblFIRST.setText("NSTRES =
0 ---> PLANE STRAIN ");
        //FIRST2 label

jlblFIRST2.setVerticalAlignment(JLabel.CENT
ER);

jlblFIRST2.setHorizontalAlignment(JLabel.LEF
T);
        jlblFIRST2.setSize(1,0);
        jlblFIRST2.setText(" / else
---> PLANE STRESS");
        //NSTRES label
jlblNSTRES.setVerticalAlignment(JLabel.CENT
```

4. JAVAによるプログラムの改良

Fortranは, Watcom Fortran Ver11.0 で行った. またJAVAはJava2 SDK, SE 1.4とNetBeans3.6を使った.

まず図-1のような簡単な2次元弾性問題の解析モデルで解説する.

図-2と図-3に示すように解析を簡単にするためや, 他のソフトより高精度な結果を出せるためプレプロセッサーを作って, プログラムが自動的に構造物を記入された有限個の要素に分解するようにした.

図-4のように COMPILER の WINDOW でこのように要素数や節点数が求まり, 要素数が多くても困らないように図も出力可能とした.

図-5, 図-6, 図-7, 図-8に示すように, 記入された勾配によって一つずつの要素厚や節点座標などをパソコンが自動的に計算するようなプレプロセッサーを開発した.

図-9や図-10 に示すように, 1要素の応力, 主応力, 1節点の変位, 反力をSWINGパッケージやGUIを利用して求めた結果はこのような表となった. また出力計算結果は以前の結果と同時に出て, かつ荷重や厚さなどの変化においても, 比較することも可能とできるようにプログラムを組んだ

また次に図-11, 図-10, 図-12 に示すように, 初期不整と残留応力度を有するI形ばりの耐荷力を求める有限変位プログラムをやはりJAVAで作り直し, Fortranとの演算時間の比較を行った.

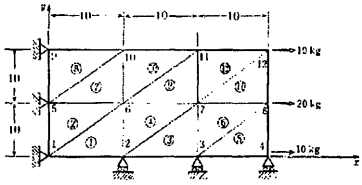


図-1 解析モデル

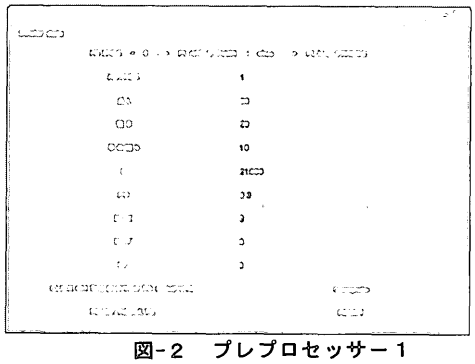


図-2 プレプロセッサー1

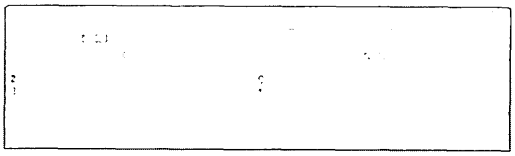


図-3 プリプロセッサー2

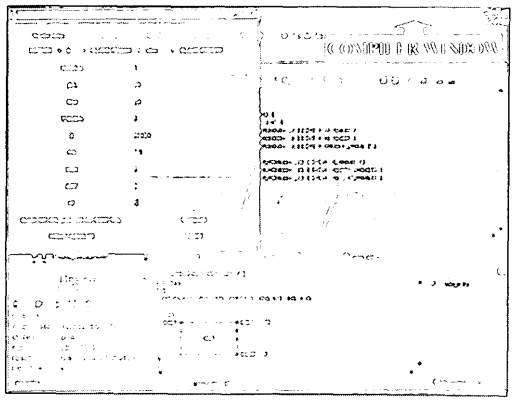


図-4 図化プログラム

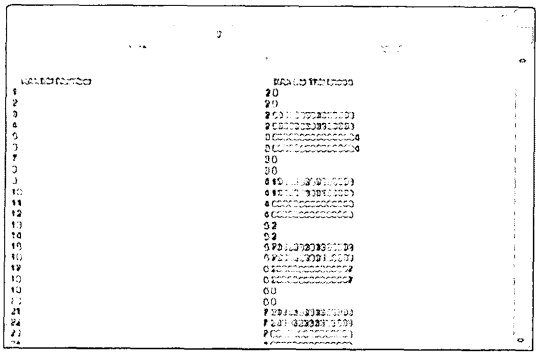


図-5 要素厚自動計算1

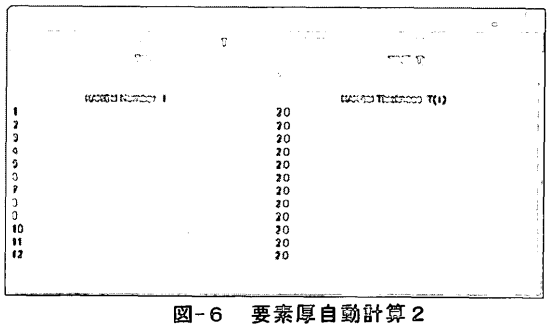


図-6 要素厚自動計算2

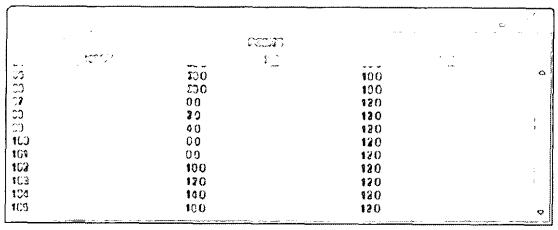


図-7 節点座標計算1

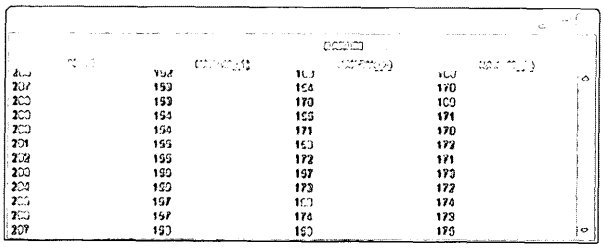
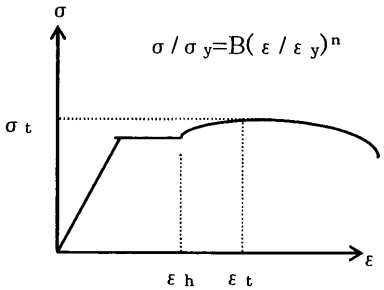


図-8 節点座標計算2

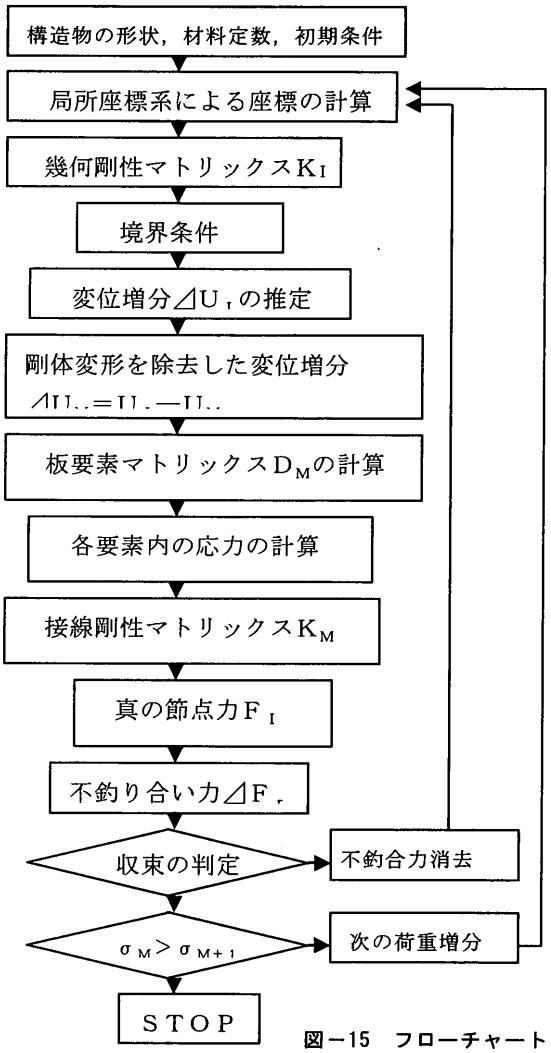
下記の図-13, 図-14に示す部材データを使用し, 図-15に示すフローチャートで, 特に浮動小数点が効いてくる不釣合い力など消去などで荷重増分の繰り返し計算を行い, FortranとJAVAの演算時間比較を行った.



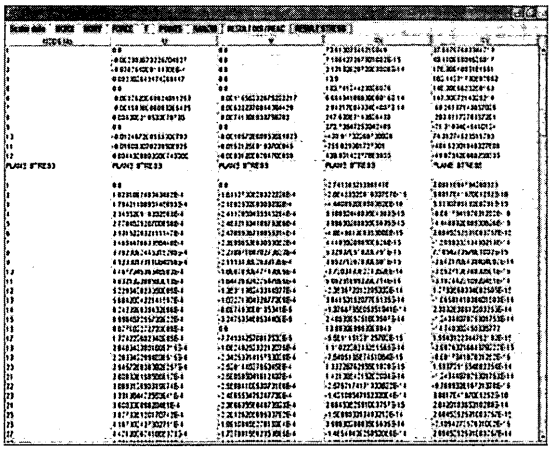
図－13　応力－ひずみ図

E	G	ν	σ_y	B
2.10E06	8.10E05	0.3	2.85E03	0.4876
n	ε_h	ε_t	k_f	k_w
0.268	0.0211	0.211	0.425	23.9

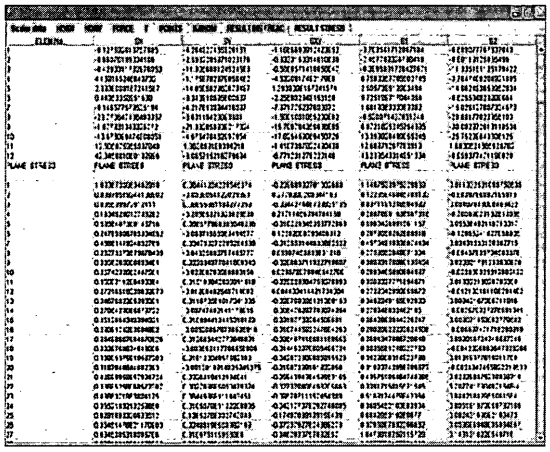
図－14　部材入力データ



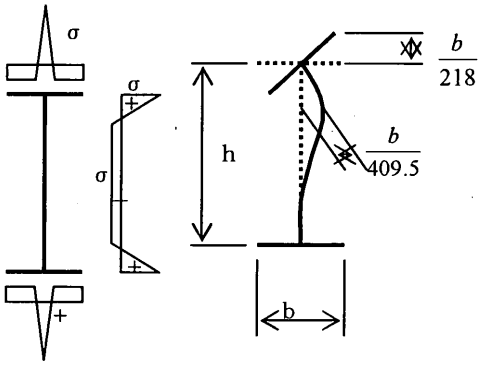
図－15　フローチャート



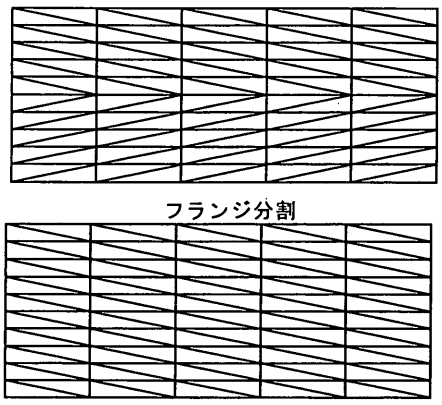
図－9　計算結果



図－10　計算結果比較



図－11　初期不整



フランジ分割

ウェブ分割

図－12　メッシュ分割

5. 結果と考察

これらの計算結果をまとめて表1に示す．表1に示すように強制変位によって有限変位の演算時間に差があるので，典型的な2つを選択して比較してみた．

ほぼ一致して，演算時間では，Fortran の方が有利であることが確認された．

しかしながら，これらの数値が，膨大なメッシュ割となるとどのような数値になるのかは，想像できないが，少なくても実行速度のみで決定できないと思われる．

それは，これから Fortran 自身が free 化してきたとは言え，ライブラリ自体は高価であり，それに比して，JAVA は，オブジェクト指向言語であり，多くの標準クラスライブラリが，充実しており，何よりも様々なプラットフォームで(JAVA VM 仮想マシーン上)作動するなど，数多くの長所を備えている．

また筆者も使用したが，NetBeans のようなコンポーネントプログラミング技術として有限要素法のライブラリを必要なものだけを組み合わせて作れるというのも長所であり，比較的楽にプログラム作成が出来た．

グラフィック関係では，JAVA に比較して Watcom Fortran や Visual Fortran などもGUIがある程度，充実しているが，なかなか使用頻度が多くなく，ソフトウェアでの共通性がない．

しかしながら，グラフィックで結果をポストプロセッサーの形で表現できえなかったことを残念に考えている．

今回は，×86 系の CPU で実施したが，AMD の CPU で実施したい．また浮動小数点に大きく影響を与える Structfp や widefp の効果も考えた比較計算も行っていきたい．

また Visual Basic や Visual C などの多くの開発環境でも実施し比較していきたいと考えている．教育的な観点からいくと，建設系の大学などでは，相変わらずプログラム開発環境として Fortran が多く，C言語や Visual Basic などは，馴染みが薄く現状では一部の浸透でしかない現状である．

逆に考えれば，多くの建設系の Fortran の財産も現在，フリーウェア化した Fortran が出現してきて始めて学生が自宅で学習・研究できる可能性も少しづつ開かれてきたと考えられが数値計算ライブラリなどは依然として高価である．

しかし PC の機種や OS に作用されない JAVA は教育的観点からも，多くのライブラリも無償であり，都合のよい開発環境であると言える．

表－1　演算時間比較

解析	RUNTIME	
	Fortran	JAVA
有限変位解析 1	42 “	67 “
有限変位解析 2	54 “	82 “
Pentium4　2.0Ghz　512MB		

参考文献

1) 内山知実：Java による連続体力学の有限要素法　森北出版
2) 赤間世紀：Java2 による数値計算　技報堂出版
3) 戸川隼人：有限要素法概論　培風館
4) 坂本衛：JavaBeans プログラミング入門－Begin the Bean－オーム社
5) Dan Brookshier ：JavaBeans デベロッパーズ・リファレンス　ソフトバンク社
6) 矢沢久雄：プログラマ養成入門講座 JAVA①　翔泳社
7) 磯山賢司：JAVA ではじめるプログラミング入門　ナツメ社