タイトル	経営科学とOR のためのWeb プログラミングによる需要予測のデータ解析
著者	福永，厚; FUKUNAGA，Atsushi
引用	北海学園大学学園論集(171): 25-38
発行日	2017-03-25

経営科学と OR のための Web プログラミングによる需要予測のデータ解析

福　永　　厚

1．はじめに

経営科学と OR（Operations Research）で扱う問題の一つに需要予測がある[1)]。需要予測は，企業のあらゆる計画の基礎になっており，生産量や仕入量，資金，生産設備，人員等の計画を決める際の重要な要因となっている。需要は，国際情勢や国内の社会情勢，政策，自然環境，市場の動向，新技術，業界，競合他社，新規参入等の様々な外部要因や企業の内部要因に依存する。

需要予測には，需要の時間的変動パターンの規則性を分析する時系列分析[2)]と，需要とその決定的要因との間に存在する関係の法則性を分析する回帰分析とがあり，本稿では回帰分析を扱う。回帰分析は統計学分野に属し，SPSS や SAS といった統計分析のパッケージソフトが数多く市販されており，それらのソフトを利用することで様々な統計分析を行うことができる。また，表計算ソフト Microsoft Excel にも統計分析に関する機能が備わっており，ある程度の統計分析が行える。これらの統計ソフトを利用するには，ソフトをコンピュータにインストールする必要があり，また，これらはローカルで動くソフトである。一方，Web ブラウザ上で回帰分析ができるようになれば，インターネット上のどこでも Web サービスとして回帰分析を行って需要予測ができるようになる。

本稿では，特別なソフトを必要とせずに，ブラウザ上でデータを入力するだけで自動的に回帰分析ができるプログラムを作成する。また，回帰分析を行う際には，説明変数と目的変数（被説明変数）との間の関係を散布図に表すことが有益であるので，自動的に散布図を作成するプログラムも作成する。Web プログラミングには JavaScript 言語を使い，散布図作成には HTML バージョン 5（以下，HTML5）の Canvas を用いる[3)]。

以下，第 2 章では需要予測における回帰分析の概要について，第 3 章では JavaScript と HTML5 の Canvas によって回帰分析と散布図の作成を自動的に行うプログラムについて解説し，さらに Excel との比較を行い，第 4 章でまとめる。

2．需要予測における回帰分析

需要予測における回帰分析[4)~6)]では，目的変数 y に売上高を取り，説明変数 x に考えられる要因を取る。説明変数が一つの場合を単回帰分析といい，2 つ以上の変数を使う場合を重回帰分析と呼ぶ。売上高 y と要因 x の

間の関係を回帰式 $y=f(x)$ で表し，関数 $f(x)$ には様々な関数が考えられる。本稿では，r 次多項式で表される関数（以下，r 次関数）を扱う。

2.1　単回帰分析

r 次関数による回帰式は，

$$y=a_0+a_1x+a_2x^2+a_3x^3+\cdots+a_rx^r \qquad (1)$$

で表される。

回帰式の係数 a_0，a_1，…，a_r を決める為には，最小 2 乗法を使う。実際に得られた n 個の x と y のデータを (x_1, y_1)，(x_2, y_2)，…，(x_n, y_n) と表し（実測値），実測値と回帰式(1)によって理論的に計算された値（予測値）との差の 2 乗 Q が，最小になるように係数 a_0，a_1，…，a_r を決める。つまり，

$$Q=\sum_{i=1}^{n}(y_i-a_0-a_1x_i-a_2x_i^2-\cdots-a_rx_i^r)^2$$

が，最小になるように係数を決める。このことは，各係数 a_j $(j=0, \cdots, r)$ が，$\dfrac{\partial Q}{\partial a_j}=0$ を満たすことを意味する。係数 a_j は，r+1 個の未知数を求める

$$\begin{cases} \sum_{i=1}^{n}1a_0+\sum_{i=1}^{n}x_ia_1+\cdots+\sum_{i=1}^{n}x_i^ra_r=\sum_{i=1}^{n}y_i \\ \qquad\qquad\vdots \\ \sum_{i=1}^{n}x_i^ra_0+\sum_{i=1}^{n}x_i^{r+1}a_1+\cdots+\sum_{i=1}^{n}x_i^{2r}a_r=\sum_{i=1}^{n}x_i^ry_i \end{cases} \qquad (2)$$

という r+1 元連立 1 次方程式を解いて求められる。

多元連立 1 次方程式を解くやり方は，一般的にはクラメルの方法を用いる。係数からつくられる r+1 次の正方行列

$$A=\begin{pmatrix} \sum_{i=1}^{n}1 & \cdots & \sum_{i=1}^{n}x_i^r \\ \vdots & \cdots & \vdots \\ \sum_{i=1}^{n}x_i^r & \cdots & \sum_{i=1}^{n}x_i^{2r} \end{pmatrix} \qquad (3)$$

の行列式が 0 でないときに，

$$a_j=\frac{1}{|A|}\begin{vmatrix} \sum_{i=1}^{n}1 & \cdots & \sum_{i=1}^{n}y_i & \cdots & \sum_{i=1}^{n}x_i^r \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \sum_{i=1}^{n}x_i^r & \cdots & \sum_{i=1}^{n}x_i^ry_i & \cdots & \sum_{i=1}^{n}x_i^{2r} \end{vmatrix} \qquad (4)$$

によって a_j が求められる。ここで，式(4)の右辺の行列式は第 j 列を，式(2)の右辺の項で置き換えた行列式である。行列 A の各要素は，実測値によって計算されるので，行列式 $|A|$ が 0 になることはない。連立方程式を解くプログラムを作成する場合，クラメルの法則を用いると行列式をいくつも計算してアルゴリズムが非常に複雑になってしまうので，効率的なアルゴリズムにする為に掃き出し法（Gauss の消去法）を用いる。

掃き出し法は，式(3)の行列 A に式(2)の右辺を列として追加した拡大係数行列 B（：で区切られた右端列に式(2)の右辺の項を挿入）を，つまり，

$$B=\begin{pmatrix} \sum_{i=1}^{n}1 & \cdots & \sum_{i=1}^{n}x_i^r & : & \sum_{i=1}^{n}y_i \\ \vdots & \cdots & \vdots & : & \vdots \\ \sum_{i=1}^{n}x_i^r & \cdots & \sum_{i=1}^{n}x_i^{2r} & : & \sum_{i=1}^{n}x_i^ry_i \end{pmatrix} \qquad (5)$$

を作成し，行列 B に行基本変形を繰り返すことで，左下方が 0 の上三角行列，すなわち，

$$\begin{pmatrix} 1 & \cdots & \cdots & \cdots & : & b_0 \\ 0 & 1 & \cdots & \cdots & : & b_1 \\ \vdots & \ddots & \ddots & \cdots & : & \vdots \\ 0 & \cdots & 0 & 1 & : & b_r \end{pmatrix} \qquad (6)$$

の形の行列に変形する。その結果，$a_r=b_r$ に

より a_r が求まり，続いて，順次 a_{r-1}，a_{r-2}，…，a_1，a_0 が求められる。このようにして，回帰係数 a_0，a_1，…，a_r が求められると，回帰式(1)を使って任意の x に対して予測値 y を計算することができる。

2.2　線形重回帰分析

要因が複数ある場合，p 個の説明変数 x_1，x_2，…，x_p による線形重回帰式は，

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_px_p \tag{7}$$

と表される。回帰係数 a_0，a_1，…，a_p は，単回帰式の場合のように，

$$\begin{cases} \sum_{i=1}^{n} 1a_0 + \sum_{i=1}^{n} x_{1i}a_1 + \cdots + \sum_{i=1}^{n} x_{pi}a_p = \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_{1i}a_0 + \sum_{i=1}^{n} x_{1i}x_{1i}a_1 + \cdots + \sum_{i=1}^{n} x_{1i}x_{pi}a_p = \sum_{i=1}^{n} x_{1i}y_i \\ \qquad\qquad \vdots \\ \sum_{i=1}^{n} x_{pi}a_0 + \sum_{i=1}^{n} x_{pi}x_{1i}a_1 + \cdots + \sum_{i=1}^{n} x_{pi}x_{pi}a_p = \sum_{i=1}^{n} x_{pi}y_i \end{cases} \tag{8}$$

という p+1 元連立 1 次方程式を解いて求められる。この方程式を解くアルゴリズムにも掃き出し法が使える。

2.3　決定係数

予測値と実測値がどのくらい合っているかを評価する指標として，決定係数 R^2 がある。

決定係数 R^2 は，

$$R^2 = \frac{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2} = 1 - \frac{\sum_{i=1}^{n} (y_i - Y_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2} \tag{9}$$

で計算される。n はデータ数，y_i は実測値，$\overline{y}$ は実測値の平均値，Y_i は予測値，$\overline{Y}$ は予測値の平均値である。

R^2 は，$0 \leqq R^2 \leqq 1$ を満たし，R^2 が 1 に近いほど予測値と実測値は合っているといえる。

2.4　Excel による回帰分析

Excel には回帰分析に関する機能が備わっている。回帰分析に関する関数には，

SLOPE…線形単回帰直線の傾き

INTERCEPT…線形単回帰直線の切片

LINEST…線形重回帰式の係数や切片等

RSQ…決定係数

がある。

Excel で単回帰分析を行う場合，x と y の実測値の散布図を作成し，プロットされたデータに対して［近似曲線の追加］を実行すると，直線や多項式，対数関数など様々な単回帰式による予測値の曲線が散布図に追加され，回帰式と決定係数を表示することができる。また，Excel のアドインにより［分析ツール］を追加することにより，［データ］タブから［分析ツール］を利用できる。分析ツールの中に回帰分析が入っており，線形重回帰式を求めることができる。

3．Web プログラミングによる需要予測のデータ解析

3.1　HTML5 の Canvas と JavaScript

Web ページを記述する HTML5（Hyper Text Markup Language Version5）では新しく Canvas 要素が導入され，JavaScript と連動することで，線や円を描くことができるようになり，グラフを作成できるようになった。Canvas 要素は，Safari，Opera，Firefox のあるバージョン以降，また，最新のブラウザ Internet Edge では対応している。

JavaScript で Canvas 要素を使うには，DOM（Document Object Model）によって，Canvas 要素を指定して操作を行う[7]。

HTML 文書中の Canvas 要素に,
<canvas id="canvas"></canvas>
というように，例えば "canvas" という ID 名をつけておく。そして，JavaScript プログラムの中で,
var c=document.getElementById("canvas");
のように，DOM の getElementById メソッドを使って，ID 名 "canvas" の部分を参照する。
var cnt=c.getContext("2d");
により，コンテキスト名を指定し，平面図形を描く際の "2d" を指定している。

本稿で使う主な Canvas 要素は以下のものである。

arc(x,y,r,0,2π,anticlockwise)…(x,y)を中心とする半径 r の円を反時計回りに描く

fill()…塗りつぶす

strokeStyle…図形の枠線の色を指定する

strokeText(t,x,y)…(x,y)から文字データ t を描く

直線を描くには，beginPath()によってパスを開始し, moveTo(x,y)で(x,y)に移動し，lineTo(x',y') で (x',y') まで線を引き，closePath()でパスを閉じ，stroke()により線を描くという手順を経る。

JavaScript の中で文字列や計算結果を表示するには，通常 document.write を用いる。しかし，本稿のように Web ページ上でフォームタグを使ってデータを入力し，同一ページに Canvas によって描画する場合に，データ解析結果を document.write で表示すると，新規に別ページが開き，そのページに解析結果のみが表示されてしまう。これを避けて最初と同じページに表示する為には，innerHTML プロパティを用いて，HTML の内容を書き換える方法を用いる[8),9)]。例えば，HTML 文書内で， のように， 要素に ID 名 "result" をつけておき，JavaScript プログラムの中で,
var result=document.getElementById ("result");
のように，DOM の getElementById メソッドを使って，ID 名 "result" の部分を参照し，
result.innerHTML= result.innerHTML+出力結果;
として，ID 名 "result" の部分に出力結果を追加していく。このような方法によって，データ解析結果が，入力テキストボックスや Canvas によるグラフと同じページに出力することができる。

3.2 単回帰分析の計算結果と Excel との比較

本稿のプログラムでは，Web ブラウザ上で需要予測のデータを入力する場合，目的変数 y は売上高であるので正の値に限定する。また，説明変数も正の値に限定し，負の値は基準値をずらすことで正の値に変換して入力するものとする。説明変数と目的変数には 3 桁までの数値を入力するとし，4 桁以上は単位を操作することで，3 桁以下に変換して入力する。プログラム自体は，4 桁以上の入力にまで拡張することは容易である。データ数 n も 10 個までと限定しているが，10 個より大きいデータ数を扱うようにプログラムを拡張することも容易である。

図 1 に単回帰分析の場合の 10 個のデータ

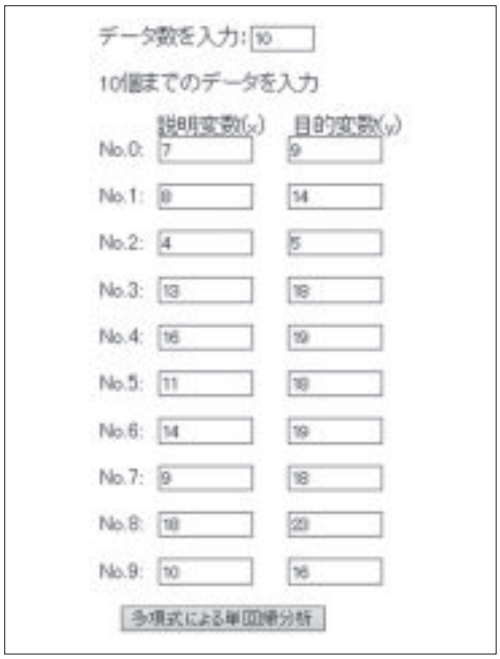

図 1　単回帰分析のデータ入力（Internet Edge によるブラウザ画面，データ数 10 個の場合）

の入力画面を Internet Edge ブラウザ画面で表示している。入力には，HTML のフォームタグの入力テキストボックスを用いている[10]。本プログラムでは，多項式の次数を任意の r 次まで対応できるようにしているが，あまり次数が大きいと図が判別しにくくなることや，奇妙なふるまい曲線になったり，高次項が効かなくなったりする場合があるので，ここでは 4 次多項式までに限定している。

図 1 の ”多項式による単回帰分析” ボタンをクリックした結果が図 2 に示されている。

1 次から 4 次までの多項式による回帰分析の結果として，各次数の回帰式の回帰係数と決定係数が表示上は四捨五入されて小数第 4 位まで表示されている（プログラム内部の計算では桁数を限定していない）。また，Canvas による散布図が描かれ，図の中に実測値と 4 次関数までの回帰式による予測値が，それぞれの回帰式を区別するために色と線のスタイルを変えて描かれている。縦軸と横軸の目盛りも，実測値の最大値を考慮し調整して描くようになっている。

図 3 には，Excel を使った場合の散布図と，［近似曲線の追加］を使用して求めた回帰式と決定係数が描かれている。本プログラムによる分析結果は，Excel の結果と一致している。決定係数は 4 次式による値が最も 1 に近くなっているが，4 次式は説明変数が実測値の範囲から外れると，他の多項式による値との間に大きな差が見られる。

図 4 にはデータ数が 6 個で目的変数が 3 桁，説明変数が 1 桁の場合のブラウザ上の入力画面と，図 5 にはブラウザ上の回帰分析結果，図 6 には Excel による分析結果が示されている。本プログラムと Excel による結果は一致している。

図 7 に本プログラムのソースが示されている。フォームタグの入力テキストボックスによりデータを入力させ，ボタンをクリックすると作成された JavaScript 関数 regression1 を実行するように作られている。関数 regression1 では，回帰分析と散布図の作成を行っている。フォームタグに入力された実測値データは，データ数は変数 n に，説明変数 x は 2 次元配列 d[i][0] に，目的変数 y は 2 次元配列 d[i][1] に格納されている。r 次関数の回帰係数を求めるために，実測値データが入った 2 次元配列 d[][] を使って，式(5)の係数拡大行列 B の各要素を計算し，各要素を 2 次元配列 m[][] に代入している。

r+1 元連立方程式を解くために作成した

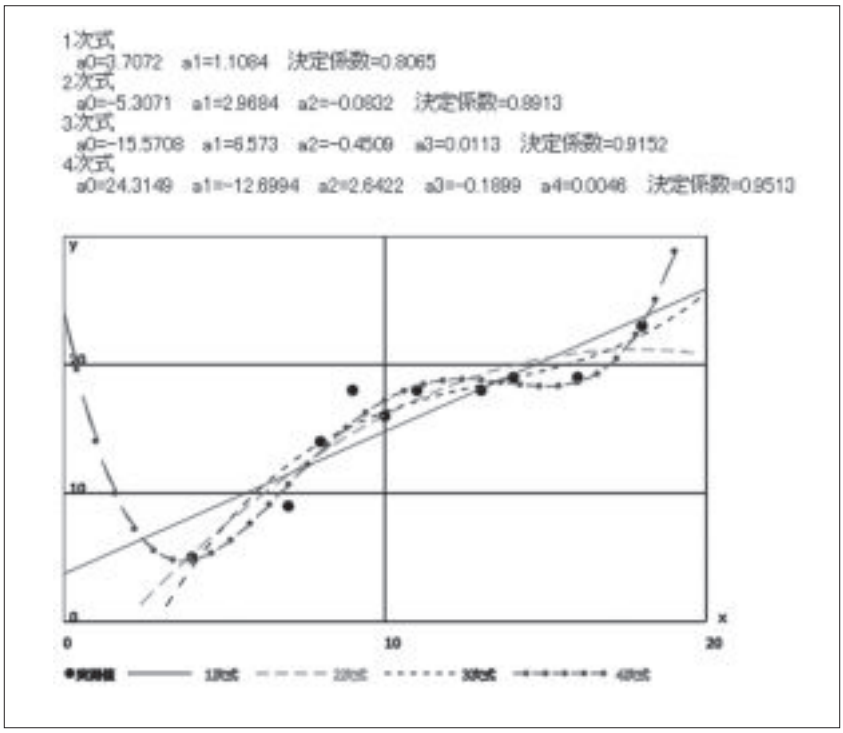

図 2　単回帰分析の結果（Internet Edge によるブラウザ画面，データ数 10 個の場合）

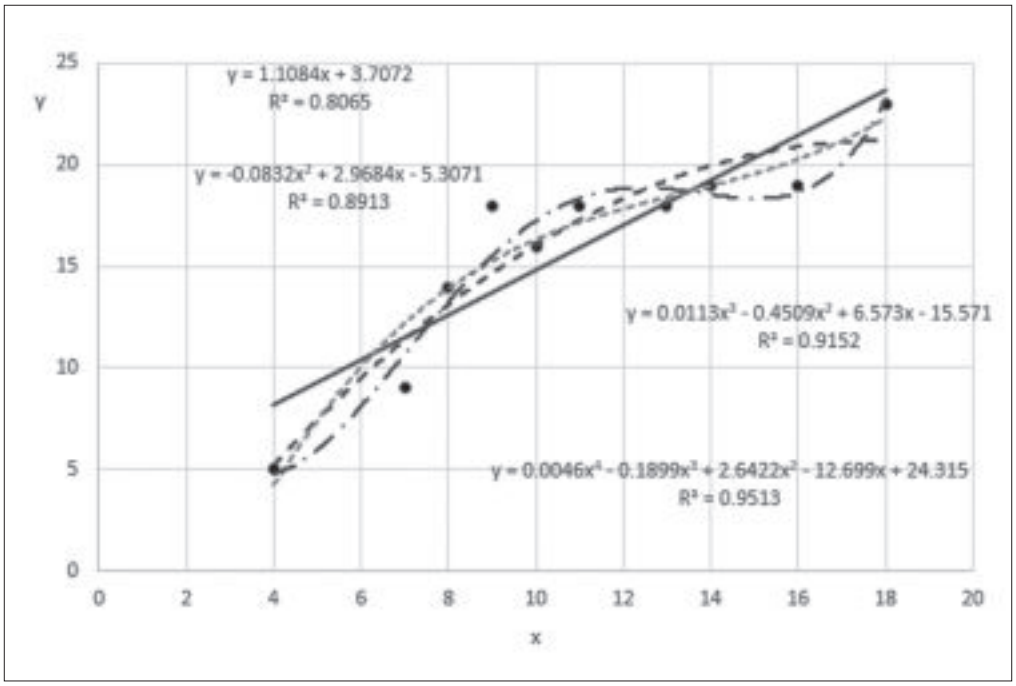

図 3　単回帰分析の結果（Excel による結果，データ数 10 個の場合）

図 4　単回帰分析のデータ入力（Internet Edge によるブラウザ画面，データ数 6 個の場合）

関数 gauss を，係数拡大行列の要素が入った２次元配列 m[][]と元の数 r+1 を引数として実行すると，m[i][r + 1](i=0, …, r)に r 次関数の第 i 項の回帰係数が計算されて返ってくるのである。この計算された回帰係数を２次元配列 a[][]に代入し，この回帰係数を使って式(1)により予測値を計算し，式(9)から決定係数を求めている。

散布図の作成では，実測値の最大値を調べ等間隔に目盛線を描く。実測値は，arc メソッドによって黒塗りの円を描いている。回帰式は，各次数ごとに描くようにし，それぞれ色と線のスタイルを変えている。

n 元連立 1 次方程式を掃き出し法によって解く関数が gauss である。n 元連立 1 次方程式の係数で作られる行列に対して行基本変形を繰り返して，式(6)のような上三角行列にし，方程式の解を求めている。

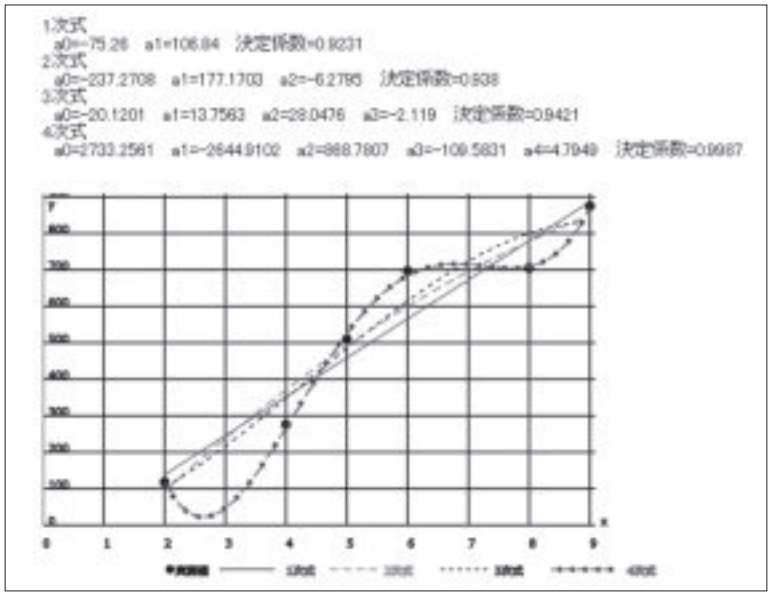

図 5　単回帰分析の結果（Internet Edge によるブラウザ画面，データ数 6 個の場合）

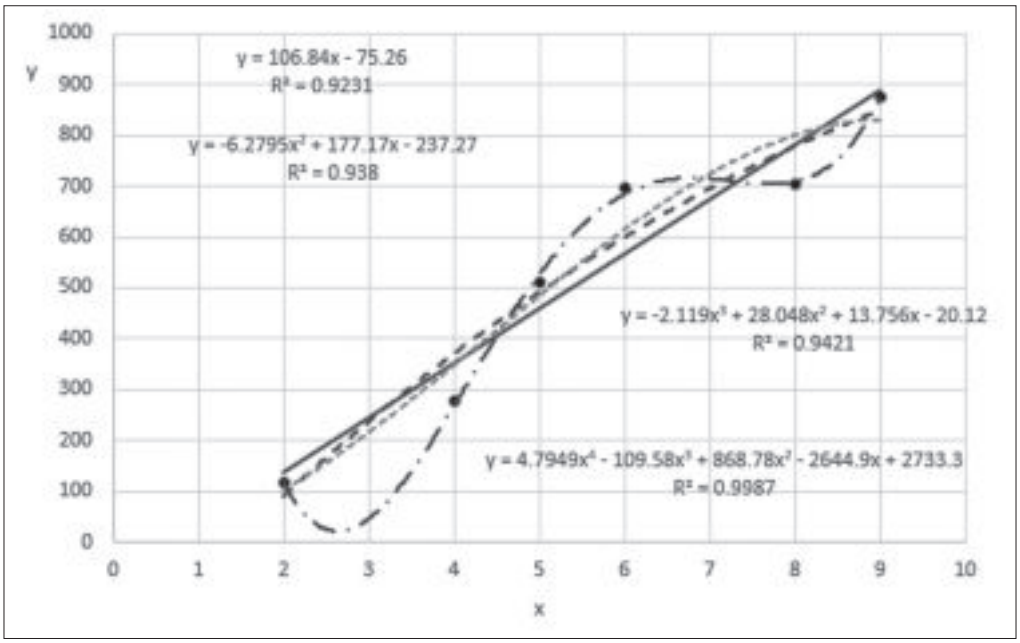

図 6　単回帰分析の結果（Excel による結果，データ数 6 個の場合）

```
<!DOCTYPE HTML>
<html lang="ja">
<head><title>多項式による単回帰分析</title></head>
<body>

<form name="form1">
データ数を入力：<input type="text" name="datasuu" size="6"><br>
<p>10個までのデータを入力</p>
　　説明変数(x)　　目的変数(y)<br>
No.0:　<input type="text" name="x0" size="10">
　<input type="text" name="y0" size="10"><br><br>
No.1:　<input type="text" name="x1" size="10">
　<input type="text" name="y1" size="10"><br><br>
No.2:　<input type="text" name="x2" size="10">
　<input type="text" name="y2" size="10"><br><br>
No.3:　<input type="text" name="x3" size="10">
　<input type="text" name="y3" size="10"><br><br>
No.4:　<input type="text" name="x4" size="10">
　<input type="text" name="y4" size="10"><br><br>
No.5:　<input type="text" name="x5" size="10">
　<input type="text" name="y5" size="10"><br><br>
No.6:　<input type="text" name="x6" size="10">
　<input type="text" name="y6" size="10"><br><br>
No.7:　<input type="text" name="x7" size="10">
　<input type="text" name="y7" size="10"><br><br>
No.8:　<input type="text" name="x8" size="10">
　<input type="text" name="y8" size="10"><br><br>
No.9:　<input type="text" name="x9" size="10">
　<input type="text" name="y9" size="10"><br><br>

<input type="button" value="多項式による単回帰分析"
 onClick="regression1();"><br><br>
</form>
<div id="result">
</div>
<canvas id="canvas" width="600" height="400"></canvas>
<br><br>

</p>

<script>
function regression1(){
var n,i,j,k,r,rmax,sum,sumy,sumyx,deter;

//rmax…多項式の最大次数、n…実測値のデータセット数
rmax=4;

n=Number(document.form1.datasuu.value);

//d[][0]…xの実測値、d[][1]…yの実測値
//m[][]…拡大係数行列、a[][]…回帰係数
var d=new Array();
 for(i=0;i<10;i++){
  d[i]=new Array();
 }

var m=new Array();
 for(i=0;i<=rmax+1;i++){
  m[i]=new Array();
}

var a=new Array();
 for(i=0;i<=rmax+1;i++){
  a[i]=new Array();
}

//フォームの実測値を2次元配列d[][]に代入
d[0][0]=Number(document.form1.x0.value);
d[1][0]=Number(document.form1.x1.value);
d[2][0]=Number(document.form1.x2.value);
d[3][0]=Number(document.form1.x3.value);
d[4][0]=Number(document.form1.x4.value);
d[5][0]=Number(document.form1.x5.value);
d[6][0]=Number(document.form1.x6.value);
d[7][0]=Number(document.form1.x7.value);
d[8][0]=Number(document.form1.x8.value);
d[9][0]=Number(document.form1.x9.value);

d[0][1]=Number(document.form1.y0.value);
d[1][1]=Number(document.form1.y1.value);
d[2][1]=Number(document.form1.y2.value);
d[3][1]=Number(document.form1.y3.value);
d[4][1]=Number(document.form1.y4.value);
d[5][1]=Number(document.form1.y5.value);
d[6][1]=Number(document.form1.y6.value);
d[7][1]=Number(document.form1.y7.value);
d[8][1]=Number(document.form1.y8.value);
d[9][1]=Number(document.form1.y9.value);

//実測値yの平均値の計算
sumy=0;
for(i=0;i<n;i++){
 sumy=sumy+d[i][1];
```

```
}
avey=sumy/n;

//r次多項式の拡大係数行列m[][]の作成

for(r=1;r<=rmax;r++){
 for(i=0;i<=r;i++){
  for(j=0;j<=r;j++){
          sum=0;
          for(k=0;k<n;k++){
          sum=sum+Math.pow(d[k][0],i+j);
          }
  m[i][j]=sum;
 }
 sum=0;
 for(k=0;k<n;k++){
  sum=sum+Math.pow(d[k][0],i)*d[k][1];
 }
 m[i][r+1]=sum;
 }

//r+1元連立1次方程式の解法
 gauss(m,r+1);

//結果の表示
var result=document.getElementById("result");

 result.innerHTML=result.innerHTML+r+"次式"+"<br>";
 for(i=0;i<=r;i++){
  a[i][r]=m[i][r+1];
  result.innerHTML=result.innerHTML+"
a"+i+"="+Math.round(10000*a[i][r])/10000+" ";
 }

//決定係数の計算
 sumy=0;
 sumyx=0;
 for(i=0;i<n;i++){
  sumy=sumy+Math.pow(d[i][1]-avey,2);
  sum=a[0][r];
  for(j=1;j<=r;j++){
          sum=sum+a[j][r]*Math.pow(d[i][0],j);
  }
  sumyx=sumyx+Math.pow(d[i][1]-sum,2);
 }
 deter=Math.round(10000*(1-sumyx/sumy))/10000;
 result.innerHTML=result.innerHTML+"　決定係数="+deter+"<br>";
}
 result.innerHTML=result.innerHTML+"<br><br>";

//散布図の作成
var d0min,d0max,d1min,d1max,xmin,xmax,ymin,ymax,xm,ym;

//実測値x,yの最大値と最小値
d0min=d[0][0];
d0max=d[0][0];
d1min=d[0][1];
d1max=d[0][1];

for(i=1;i<n;i++){
 if(d[i][0]<d0min){
  d0min=d[i][0];
 }
 if(d[i][1]<d1min){
  d1min=d[i][1];
 }
 if(d[i][0]>d0max){
  d0max=d[i][0];
 }
 if(d[i][1]>d1max){
  d1max=d[i][1];
 }
}

//実測値x,yの最大目盛りとスケール
if(d0max<10){
 xm=1;
 xmax=Math.ceil(d0max);
 xmin=Math.floor(d0min);
}
else if(d0max<100){
 xm=10;
 xmax=Math.ceil(d0max/10);
 xmin=Math.floor(d0min/10);
}
else if(d0max<1000){
 xm=100;
 xmax=Math.ceil(d0max/100);
 xmin=Math.floor(d0min/100);
}
else{
 result.innerHTML=result.innerHTML+" xのデータは、正で1000より小さい値で
入力<br>";
}

if(d1max<10){
 ym=1;
 ymax=Math.ceil(d1max);
 ymin=Math.floor(d1min);
}
else if(d1max<100){
 ym=10;
 ymax=Math.ceil(d1max/10);
 ymin=Math.floor(d1min/10);
}
else if(d1max<1000){
 ym=100;
 ymax=Math.ceil(d1max/100);
 ymin=Math.floor(d1min/100);
}
else{
 result.innerHTML=result.innerHTML+" y のデータは、正で1000より小さい値で
入力<br>";
}

var c=document.getElementById("canvas");
var cnt=c.getContext("2d");

var x0,y0,xf,yf,w,h,xw,yw,xx;
x0=0;
y0=0;
w=500;
h=300;
xf=x0+w;
yf=y0+h;

xw=Math.floor(w/xmax);
for(i=0;i<=xmax;i++){
 cnt.beginPath();
 cnt.moveTo(x0+xw*i,y0+yf);
 cnt.lineTo(x0+xw*i,y0);
 cnt.closePath();
 cnt.stroke();
 cnt.strokeText(xm*i,x0+xw*i,y0+yf+20);
}
 cnt.strokeText("x",x0+xw*xmax+10,y0+yf);

yw=Math.floor(h/ymax);
for(i=0;i<=ymax+1;i++){
 cnt.beginPath();
 cnt.moveTo(x0,yf-yw*i);
 cnt.lineTo(x0+xf,yf-yw*i);
 cnt.closePath();
 cnt.stroke();
 cnt.strokeText(ym*i,x0+5,yf-yw*i);
}
 cnt.strokeText("y",x0+5,yf-yw*ymax+10);

for(i=0;i<n;i++){
 cnt.beginPath();
 cnt.arc(d[i][0]*xw/xm,yf-d[i][1]*yw/ym, 4, 0, Math.PI*2, false);
 cnt.fill();
 cnt.closePath();
 cnt.stroke();
}
 cnt.arc(xmin*xw+5,yf+40, 4, 0, Math.PI*2, false);
 cnt.fill();
 cnt.strokeText("実測値",xmin*xw+10,yf+45);

//1次式の描画
cnt.beginPath();
cnt.moveTo(xmin*xw,yf-(a[0][1]+a[1][1]*xmin*xm)*yw/ym);
cnt.strokeStyle="rgb(255,0,0)";
cnt.lineTo(xmax*xw,yf-(a[0][1]+a[1][1]*xmax*xm)*yw/ym);
cnt.moveTo(xmin*xw+50,yf+40);
cnt.lineTo(xmin*xw+100,yf+40);
cnt.closePath();
cnt.stroke();
cnt.strokeText("1次式",xmin*xw+110,yf+45);

//2次式の描画
var ten=new Array();
var liner=new Array();

xx=(xmax-xmin)/100;
for(i=0;i<=100;i++){
 ten[i]=Math.floor(1000*(xmin+xx*i))/1000;
}
```

```
for(i=0;i<=100;i++){
 sum=a[0][2];
  for(j=1;j<=2;j++){
        sum=sum+a[j][2]*Math.pow(ten[i]*xm,j);
  }
 liner[i]=sum;
}
for(i=0;i<100;i=i+3){
 if(liner[i]*yw/ym>0){
  cnt.beginPath();
  cnt.moveTo(ten[i]*xw,yf-liner[i]*yw/ym);
  cnt.strokeStyle="rgb(0,255,0)";
  cnt.lineTo(ten[i+2]*xw,yf-liner[i+2]*yw/ym);
  cnt.closePath();
  cnt.stroke();
 }
}
for(i=0;i<10;i=i+3){
 cnt.beginPath();
 cnt.moveTo(ten[i]*xw+150,yf+40);
 cnt.lineTo(ten[i+2]*xw+150,yf+40);
 cnt.closePath();
 cnt.stroke();
}
 cnt.strokeText("2次式",ten[10]*xw+160,yf+45);

//3次式の描画
for(i=0;i<=100;i++){
 sum=a[0][3];
 for(j=1;j<=3;j++){
  sum=sum+a[j][3]*Math.pow(ten[i]*xm,j);
 }
 liner[i]=sum;
}
for(i=0;i<100;i=i+2){
 if(liner[i]*yw/ym>0){
  cnt.beginPath();
  cnt.moveTo(ten[i]*xw,yf-liner[i]*yw/ym);
  cnt.strokeStyle="rgb(0,0,255)";
  cnt.lineTo(ten[i+1]*xw,yf-liner[i+1]*yw/ym);
  cnt.closePath();
  cnt.stroke();
 }
}
for(i=0;i<12;i=i+2){
 cnt.beginPath();
 cnt.moveTo(ten[i]*xw+250,yf+40);
 cnt.lineTo(ten[i+1]*xw+250,yf+40);
 cnt.closePath();
 cnt.stroke();
}
 cnt.strokeText("3次式",ten[10]*xw+260,yf+45);

//4次式の描画
for(i=0;i<=100;i++){
  sum=a[0][4];
        for(j=1;j<=4;j++){
        sum=sum+a[j][4]*Math.pow(ten[i]*xm,j);
        }
  liner[i]=sum;
}
for(i=0;i<100;i=i+3){
  if(liner[i]*yw/ym>0){
        cnt.beginPath();
        cnt.moveTo(ten[i]*xw,yf-liner[i]*yw/ym);
        cnt.strokeStyle="rgb(255,0,255)";
        cnt.lineTo(ten[i+1]*xw,yf-liner[i+1]*yw/ym);
        cnt.arc(ten[i+2]*xw,yf-liner[i+2]*yw/ym, 2, 0, Math.PI*2, false);
        cnt.fill();
        cnt.closePath();
        cnt.stroke();

  }
}
for(i=0;i<14;i=i+3){
  cnt.beginPath();
  cnt.moveTo(ten[i]*xw+350,yf+40);
  cnt.lineTo(ten[i+1]*xw+350,yf+40);
  cnt.arc(ten[i+2]*xw+350,yf+40, 2, 0, Math.PI*2, false);
  cnt.fill();
  cnt.closePath();
  cnt.stroke();
}
 cnt.strokeText("4次式",ten[10]*xw+380,yf+45);

}

//n元連立1次方程式を掃き出し法によって解く関数
function gauss(m,n){
var i,j,k,mii,mji;

for(i=0;i<n;i++){
 mii=m[i][i];
  for(j=0;j<n+1;j++){
  m[i][j]=m[i][j]/mii;
  }
 for(j=i+1;j<n;j++){
  mji=m[j][i];
        for(k=i;k<n+1;k++){
        m[j][k]=m[j][k]-mji*m[i][k];
        }
 }
}

for(i=n-1;i>0;i--){
 for(j=i-1;j>=0;j--){
  mji=m[j][i];
        for(k=i;k<n+1;k++){
        m[j][k]=m[j][k]-mji*m[i][k];
        }
 }
}

return m;
}

</script>
</body>
</html>
```

図 7　単回帰分析のプログラムソース

3.3　線形重回帰分析の計算結果

2 変数の線形重回帰分析の入力と出力結果の Internet Edge ブラウザ画面が図 8 に示されている。重回帰分析の場合は平面図では表

図 8　2 変数線形重回帰分析のデータ入力と分析結果（Internet Edge によるブラウザ画面）

せないので散布図は作成していない。Excel のデータ分析の回帰分析による結果が図 9 に示されている。図 9 の表に示されている係数の切片が a_0，X 値 1 が a_1，X 値 2 が a_2 に，重決定 R2 が決定係数に対応している。本プログラムによる回帰係数と決定係数（表示上は小数第 4 位まで表示している）と Excel による結果は一致している。3 変数の線形重回帰分析の入力と出力結果の Internet Edge ブラウザ画面が図 10 に，Excel のデータ分析の回帰分析による結果が図 11 に示されている。回帰係数と決定係数は一致している。

図 12 に，3 変数線形重回帰分析のソースプログラムを示す。重回帰分析の場合の式(8)は，単回帰分析の場合の式(2)と係数が異なるだけで，同様な方法で解ける。従って，拡大係数行列を 3 変数重回帰分析用に設定すれば，多元連立 1 次方程式を解く関数 gauss が使える。

ここでは，3 変数までの結果を表示したが，さらに多い変数の場合にもプログラムは拡張可能である。

4．おわりに

本稿では，JavaScript と HTML5 の Canvas を使って，需要予測のデータ解析と散布図の作成を行った。単回帰分析の場合，4 次関数までの回帰式の回帰係数と決定係数を求め，求めた値は Excel を使って計算した値と一致していた。散布図の自動的作成では，実測値と予測値が適当な位置に描かれていた。また，3 変数までの線形重回帰分析を

目的変数(y)	説明変数1(x_1)	説明変数2(x_2)
212	72	6.7
75	52	14.7
285	65	8.1
449	41	4.6
112	95	9.8
183	86	6.4
306	39	7.6

概要

回帰統計	
重相関 R	0.953220925
重決定 R2	0.908630132
補正 R2	0.862945198
標準誤差	47.40384403
観測数	7

分散分析表

	自由度	変動	分散	された分	有意 F
回帰	2	89386.35943	44693.18	19.889	0.006348
残差	4	8988.497716	2247.1244		
合計	6	98374.85714			

	係数	標準誤差	t	P-値	下限 95%	上限 95%	下限 95.0%	上限 95.0%
切片	689.9897756	76.29475262	9.0437383	0.0008	478.1616	901.818	478.1616	901.818
X 値 1	-3.326341108	0.89543578	-3.714774	0.0206	-5.81247	-0.84021	-5.81247	-0.84021
X 値 2	-29.37953248	5.949490217	-4.93816	0.0078	-45.898	-12.8611	-45.898	-12.8611

図 9　2 変数線形重回帰分析のデータ入力と分析結果（Excel のデータ分析ツールを使った場合）

図 10　3 変数線形重回帰分析のデータ入力と分析結果（Internet Edge によるブラウザ画面）

行い，回帰係数は Excel による計算と一致していた。

今後の課題として，単回帰分析の場合の散布図は，入力するデータによっては予測値が描画領域をはみ出す場合があり，より精巧化が必要となる。回帰分析は，単に決定係数を求めて一致度を見るだけでなく，分散分析や有意性など様々な統計的な扱いが必要とされる。単回帰分析の場合には，多項式関数のみならず, 対数関数や累乗関数, 指数関数を使った場合も必要となる。重回帰分析の場合，1 次関数のみならず r 次関数を使えるように拡張する。

r 次多項式による p 変数重回帰モデルは，

目的変数(y)	説明変数1(x_1)	説明変数2(x_2)	説明変数3(x_3)
72	68	74	61
70	72	72	68
84	71	63	67
45	44	33	31
47	64	57	54
59	61	76	66
64	71	73	64
45	44	55	51
55	46	50	49
86	87	77	93

概要

回帰統計	
重相関 R	0.904567027
重決定 R2	0.818241506
補正 R2	0.727362259
標準誤差	8.942973028
観測数	10

分散分析表

	自由度	変動	分散	観測された分散比	有意 F
回帰	3	2160.2394	720.0798001	9.0036	0.012205
残差	6	479.8605995	79.97676659		
合計	9	2640.1			

	係数	標準誤差	t	P-値	下限 95%	上限 95%	下限 95.0%	上限 95.0%
切片	6.716714602	14.83614544	0.452726392	0.6667	-29.586	43.01945	-29.586	43.01945
説明変数1(x1)	0.581318218	0.474603956	1.22484908	0.2665	-0.58	1.742632	-0.58	1.742632
説明変数2(x2)	-0.432706382	0.41480058	-1.043167254	0.3371	-1.44769	0.582274	-1.44769	0.582274
説明変数3(x3)	0.790947736	0.498147107	1.586574982	0.1637	-0.42857	2.00927	-0.42857	2.00927

図 11　3 変数線形重回帰分析のデータ入力と分析結果（Excel のデータ分析ツールを使った場合）

```
<!DOCTYPE HTML>
<html lang="ja">
<head><title>3変数線形重回帰分析</title></head>
<body>

<form name="form1">
データ数を入力：<input type="text" name="datasuu" size="6"><br>
<p>10個までのデータセットを入力</p>
　　目的変数(y) 説明変数１(x1) 説明変数２(x2) 説明変数３(x3)<br>
No.0:　<input type="text" name="y0" size="10">
　<input type="text" name="x10" size="10">
　<input type="text" name="x20" size="10">
　<input type="text" name="x30" size="10"><br><br>
No.1:　<input type="text" name="y1" size="10">
　<input type="text" name="x11" size="10">
　<input type="text" name="x21" size="10">
　<input type="text" name="x31" size="10"><br><br>
No.2:　<input type="text" name="y2" size="10">
　<input type="text" name="x12" size="10">
　<input type="text" name="x22" size="10">
　<input type="text" name="x32" size="10"><br><br>
No.3:　<input type="text" name="y3" size="10">
　<input type="text" name="x13" size="10">
　<input type="text" name="x23" size="10">
　<input type="text" name="x33" size="10"><br><br>
No.4:　<input type="text" name="y4" size="10">
　<input type="text" name="x14" size="10">
　<input type="text" name="x24" size="10">
　<input type="text" name="x34" size="10"><br><br>
No.5:　<input type="text" name="y5" size="10">
　<input type="text" name="x15" size="10">
　<input type="text" name="x25" size="10">
　<input type="text" name="x35" size="10"><br><br>
No.6:　<input type="text" name="y6" size="10">
　<input type="text" name="x16" size="10">
　<input type="text" name="x26" size="10">
　<input type="text" name="x36" size="10"><br><br>
No.7:　<input type="text" name="y7" size="10">
　<input type="text" name="x17" size="10">
　<input type="text" name="x27" size="10">
　<input type="text" name="x37" size="10"><br><br>
No.8:　<input type="text" name="y8" size="10">
　<input type="text" name="x18" size="10">
　<input type="text" name="x28" size="10">
　<input type="text" name="x38" size="10"><br><br>
No.9:　<input type="text" name="y9" size="10">
　<input type="text" name="x19" size="10">
　<input type="text" name="x29" size="10">
　<input type="text" name="x39" size="10"><br><br>

<input type="button" value="3変数線形重回帰分析"
 onClick="regression3();"><br><br>
</form>
<div id="result">
</div>
<canvas id="canvas" width="600" height="400"></canvas>
<br><br>

</p>

<script>
function regression3(){
var n,i,j,k,p,pmax,sum,sumy,sumyx,deter;

//pmax…説明変数の数、n…実測値のデータセット数
n=Number(document.form1.datasuu.value);
pmax=3;

var d=new Array();
 for(i=0;i<10;i++){
  d[i]=new Array();
 }

var m=new Array();
 for(i=0;i<=pmax+1;i++){
  m[i]=new Array();
}

var a=new Array();

//フォームの実測値を2次元配列d[][]に代入
//2次元配列d[][1]～d[][3]がx1～x3、d[][4]がy
d[0][1]=Number(document.form1.x10.value);
d[1][1]=Number(document.form1.x11.value);
d[2][1]=Number(document.form1.x12.value);
d[3][1]=Number(document.form1.x13.value);
d[4][1]=Number(document.form1.x14.value);
d[5][1]=Number(document.form1.x15.value);
d[6][1]=Number(document.form1.x16.value);
d[7][1]=Number(document.form1.x17.value);
d[8][1]=Number(document.form1.x18.value);
d[9][1]=Number(document.form1.x19.value);

d[0][2]=Number(document.form1.x20.value);
d[1][2]=Number(document.form1.x21.value);
d[2][2]=Number(document.form1.x22.value);
d[3][2]=Number(document.form1.x23.value);
d[4][2]=Number(document.form1.x24.value);
d[5][2]=Number(document.form1.x25.value);
d[6][2]=Number(document.form1.x26.value);
d[7][2]=Number(document.form1.x27.value);
d[8][2]=Number(document.form1.x28.value);
d[9][2]=Number(document.form1.x29.value);

d[0][3]=Number(document.form1.x30.value);
d[1][3]=Number(document.form1.x31.value);
d[2][3]=Number(document.form1.x32.value);
d[3][3]=Number(document.form1.x33.value);
d[4][3]=Number(document.form1.x34.value);
d[5][3]=Number(document.form1.x35.value);
d[6][3]=Number(document.form1.x36.value);
d[7][3]=Number(document.form1.x37.value);
d[8][3]=Number(document.form1.x38.value);
d[9][3]=Number(document.form1.x39.value);

d[0][4]=Number(document.form1.y0.value);
d[1][4]=Number(document.form1.y1.value);
d[2][4]=Number(document.form1.y2.value);
d[3][4]=Number(document.form1.y3.value);
d[4][4]=Number(document.form1.y4.value);
d[5][4]=Number(document.form1.y5.value);
d[6][4]=Number(document.form1.y6.value);
d[7][4]=Number(document.form1.y7.value);
d[8][4]=Number(document.form1.y8.value);
d[9][4]=Number(document.form1.y9.value);

//a0の係数1をd[i][0]に設定
for(i=0;i<n;i++){
 d[i][0]=1;
}

///実測値yの平均値の計算
sumy=0;
for(i=0;i<n;i++){
 sumy=sumy+d[i][pmax+1];
}
avey=sumy/n;

//pmax個の説明変数の拡大係数行列m[][]の作成
for(p=0;p<=pmax;p++){
 for(i=0;i<=pmax;i++){
  sum=0;
  for(j=0;j<n;j++){
          sum=sum+d[j][p]*d[j][i];
  }
  m[p][i]=sum;
 }
 sum=0;
 for(j=0;j<n;j++){
 sum=sum+d[j][p]*d[j][pmax+1];
 }
 m[p][pmax+1]=sum;
}

//pmax+1元連立1次方程式の解法
 gauss(m,pmax+1);

var result=document.getElementById("result");

 for(i=0;i<=pmax;i++){
  a[i]=m[i][pmax+1];
  result.innerHTML=result.innerHTML+"
a"+i+"="+Math.round(10000*a[i])/10000+" ";
 }

//決定係数の計算

 sumy=0;
 sumyx=0;
 for(i=0;i<n;i++){
  sumy=sumy+Math.pow(d[i][pmax+1]-avey,2);
  sum=a[0];
  for(j=1;j<=pmax;j++){
          sum=sum+a[j]*d[i][j];
  }
  sumyx=sumyx+Math.pow(d[i][pmax+1]-sum,2);
 }
 deter=Math.round(10000*(1-sumyx/sumy))/10000;
```

```
 result.innerHTML=result.innerHTML+"　決定係数="+deter+"<br>";

 result.innerHTML=result.innerHTML+"<br><br>";

}

//n元連立1次方程式を掃き出し法によって解く関数
function gauss(m,n){
var i,j,k,mii,mji;

for(i=0;i<n;i++){
 mii=m[i][i];
  for(j=0;j<n+1;j++){
  m[i][j]=m[i][j]/mii;
  }
 for(j=i+1;j<n;j++){
  mji=m[j][i];
        for(k=i;k<n+1;k++){
        m[j][k]=m[j][k]-mji*m[i][k];
        }
 }
}

for(i=n-1;i>0;i--){
 for(j=i-1;j>=0;j--){
  mji=m[j][i];
        for(k=i;k<n+1;k++){
        m[j][k]=m[j][k]-mji*m[i][k];
        }
 }
}

return m;
}

</script>
</body>
</html>
```

図 12　3 変数線形重回帰分析のプログラムソース

$$y = a_0 + a_{11}x_1 + a_{12}x_1^{\,2} + \cdots + a_{1r}x_1^{\,r} + a_{21}x_2 + \cdots + a_{2r}x_2^{\,r} + \cdots + a_{p1}x_p + \cdots + a_{pr}x_p^{\,r}$$

で表され，最小二乗法によって回帰係数 a_0，a_{11}，…，a_{1r}，…，a_{p1}，…，a_{pr} が，pr＋1 元連立 1 次方程式を解くことによって求められる。多元連立 1 次方程式は，2.1 で述べた掃き出し法によって求められるので，多項式非線形重回帰分析も行えることになる。また，多項式だけではなく，対数関数や累乗関数，指数関数は，多項式と似た扱いができるので，それらの関数を使った重回帰分析も行える。さらに，回帰分析だけでなく時系列分析も扱うようにすることが必要となる。

参 考 文 献

1）宮川公男：「経営情報入門」実教出版，1999 年

2）福永厚：「経営科学のための Java プログラミングによる需要予測のデータ解析」，北海学園大学開発論集第 69 号，pp.111-122，2002 年

3）福永厚：「経営科学と OR のための Web プログラミングによる窓口サービスのデータ解析」，北海学園大学学園論集第 170 号，pp.17-27，2016 年

4）田中豊，脇本和昌「多変量統計解析法」，現代数学社，1990 年

5）木下栄蔵「多変量解析入門」，啓学出版，1992 年

6）石村貞夫「すぐわかる多変量解析入門」，東京図書，1993 年

7）高橋麻奈：「やさしい JavaScript の基本」，SB クリエイティブ，2014 年

8）伊藤静香：「3 日でマスター JavaScript」，ソシム，2014 年

9）河西朝雄：「ゼロからわかる JavaScript 超入門」，技術評論社，2010 年

10）福永厚：「情報教育のための Web プログラミングによるアンケート作成とデータ解析について」，北海学園大学学園論集第 169 号，pp.17-25，2016 年