タイトル	Webプログラミングによる2変数線形計画最大化問題のグラフ解法
著者	福永，厚；FUKUNAGA，Atsushi
引用	北海学園大学学園論集(184)：65-73
発行日	2021-03-25

Webプログラミングによる
2変数線形計画最大化問題のグラフ解法

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線形計画問題がある[1)]。線形計画問題とは，線形な不等式で表される制約条件のもとで，線形で表される目的関数を最大化あるいは最小化する問題である。経営においては，限られた資源や設備，人材などの制約条件の下で利益を最大化したり，コストを最小化するような配分を求める配分問題が線形計画問題として扱われる。

線形計画問題を解く線形計画法（Linear Programming：LP）には，一般的にはシンプレックス法と呼ばれる方法がある。シンプレックス法では，最初にシンプレックス表を作成し，シンプレックス基準の値を計算してその値を改善するように，表を何度も変更していきながら最適解を見出していく。シンプレックス法は変数が多い場合にも対応していて，一般的な場合に最適解を求めることができるが，何度も表の改善を行うという多大な労力を必要とし，また，解法のアルゴリズムが難解で手順の意味がつかみにくい。

Microsoft社の表計算ソフトExcelにはソルバーという機能があり，ソルバーに制約条件式や目的関数の式を適切に入力すると，変数が多い場合にも自動的に線形計画問題の最適解を見出すことができる。ソルバーを用いれば，式の入力作業に手間はかかるが，式を入力してしまえばコンピュータが自動的に最適解を見つけ出してくれる。

シンプレックス法やソルバーは一般的な場合に最適解を見出してくれる方法であるが，見つけ出す途中の過程が見えずブラックボックスになってしまう。最適解を見つけ出すだけならばそれで良いが，問題に対する深い理解や解の意義についての知見が得られにくい。

線形計画問題を扱う通常のテキストでは，初めはグラフによる解法から説明されていく。グラフによる解法は，線形計画問題の変数が2つの場合に，2つの変数x, yを横軸（x軸）と縦軸（y軸）に取り，制約条件式を満たす領域を平面グラフに表し，目的関数の式を平行移動しながら最適解を見出していく方法である。グラフによる解法では，制約条件式を満たす領域と目的関数の式の関係から，目的関数を最大化または最小化する解をグラフから読み取ることにより，問題に対する深い理解と最適解の意義を捉えることができ，非常に教育的である。グラフによる解法は2変数の場合の2次元平面グラフと3変数の場合

の 3 次元空間グラフに限定され，グラフを描く労力がかかる。

そこで，本稿では 2 変数の線形計画の最大化問題に限って，制約条件式と目的関数の式を入力するだけで, 自動的にグラフを作成し，最適解を求めるプログラムを作成する。グラフ作成においては，HTML（HyperText Markup Language）のバージョン 5（以下，HTML5）の Canvas 要素とプログラミング言語の JavaScript の連携[2)~4)] により，Web ブラウザ上に表示できるようにする。HTML5 と JavaScript を用いると，Excel のような特別なソフトウェアを必要とせずに，ブラウザ上で必要なデータを入力するだけで自動的にグラフを作成し，最適解を求めることができる。

これにより，それぞれの制約条件式がどのように関わっているのかがグラフからわかり，制約条件式や目的関数を変えてみた場合にどのように解が変化するかを見ることができるようになる。

以下，第 2 章では線形計画の最大化問題のグラフによる解法について述べ，第 3 章では HTML5 の Canvas と JavaScript による線形計画の最大化問題の自動グラフ作成と最適解を求めるプログラムおよび実行結果について論述し，第 4 章でまとめる。

2．線形計画問題のグラフによる解法

2.1　最大化問題

線形計画の最大化問題では，線形な不等式で表される制約条件のもとで，線形で表される目的関数を最大化する。

例として，次のような場合を挙げる。ある工場では 3 種類の原料 1，原料 2，原料 3 を使って，2 種類の製品 X，Y を製造している。製品 X を 1 単位製造する為には原料 1 を 1 単位，原料 2 を 1 単位，原料 3 を 2 単位必要とする。製品 Y を 1 単位製造する為には原料 1 を 1 単位，原料 2 を 2 単位，原料 3 を 1 単位必要とする。原料 1 は 8 単位まで，原料 2 は 14 単位まで，原料 3 は 13 単位までしか使えない。製品 X の 1 単位当たりの利益は 2 万円，製品 Y は 3 万円である。このとき，製品 X と製品 Y をどれだけ作ると最も大きい利益が得られるかを求めることが線形計画の最大化問題となる。

まず，製品 X を x 単位，製品 Y を y 単位製造するものとする。ただし，x と y は $x \geqq 0$，$y \geqq 0$ を満たす実数とする。これらの条件を表に表すと，表 1 のようになる。

制約条件を数式で表すと，

$x \geqq 0$，$y \geqq 0$

$x + y \leqq 8$

$x + 2y \leqq 14$

$2x + y \leqq 13$

となる。

目的関数(利益)z は，

$z = 2x + 3y$　…①

と表され，制約条件を満たしながら，z が最

表 1　例題の条件一覧

	製品 X(x)	製品 Y(y)	使用できる原料の上限
原料 1 の必要量（1 単位製造当たり）	1 単位	1 単位	8 単位
原料 2 の必要量（1 単位製造当たり）	1 単位	2 単位	14 単位
原料 3 の必要量（1 単位製造当たり）	2 単位	1 単位	13 単位
利益（1 単位製造当たり）	2 万円	3 万円	

大となるような x,y が最適解となる。

制約条件を満たす領域をグラフ上に描くときは，制約条件式の等号の場合である境界直線

$x+y=8$　　…②

$x+2y=14$　…③

$2x+y=13$　…④

を描く必要がある。

図 1 に，境界直線と制約条件を満たす領域（斜線）を示す。$x \geqq 0$，$y \geqq 0$ によりグラフは xy 平面の第 1 象限に限定し，3 つの境界直線②～④を描き，境界直線上を含みかつそれよりも下の領域が不等式を満たす領域となる。3 つの不等式すべてを同時に満たす領域は，斜線を引いた五角形の内側の部分で五角形の周も含む。

直線の傾きが最もなだらかな境界直線③と y 軸との交点を A，境界直線②と③の交点を B，境界直線②と④の交点を C，直線の傾きが最も急な境界直線④と x 軸との交点を D とすると，制約条件を満たす領域は，原点を O として，五角形 OABCD の周を含んだ内側となる。

この問題をグラフによって解く場合には，境界直線の傾きと目的関数の式の傾きの関係を調べる。境界直線②の傾きは $y=-x+8$ と変形して -1，③の傾きは $-\frac{1}{2}$，④の傾きは -2 となる。目的関数の式①の傾きは，式①を変形して，

$$y=-\frac{2}{3}x+\frac{1}{3}z \quad \cdots⑤$$

により，$-\frac{2}{3}$である。

傾きは，$-2<-1<-\frac{2}{3}<-\frac{1}{2}$ の関係にあり，目的関数の式⑤の傾きは②よりなだらか

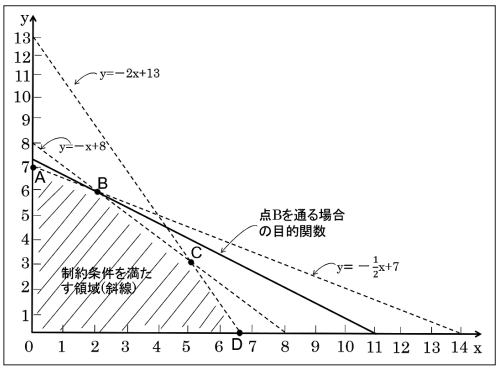

図 1　例題のグラフ

で③より急で，②と③の間にあることがわかる。目的関数 z の最大値を求める為には，式⑤が制約条件を満たしながら，y 軸と交わる y- 切片の値を最大化すれば良いので，目的関数の式⑤を平行移動しながら，y 軸と最も高い位置で交わるには，図 1 より⑤が点 B を通るときであることが見て取れる。

点 B は②と③の交点であることから，②と③の連立方程式を解くことにより，点 B の座標 (2, 6) が求められる。

したがって，$x=2$，$y=6$ のとき，最大利益 $z=2*2+3*6=22$ が求められ，この問題の最適解は，製品 X を 2 単位，製品 Y を 6 単位製造するとき，最大利益 22 万円が得られるというものになる。

2.2　HTML5 の Canvas と JavaScript

HTML は，Web ページを記述する言語であり，W3C によって 2014 年に新しいバージョンである HTML バージョン 5 に改定された[5)~7)]。

HTML5 では，新しく Canvas 要素が導入され，JavaScript と連動することにより，画像やアニメーションの動的コンテンツが生成

できるようになった。

Canvas 要素は，元々はアップル社が Mac OS に導入した技術で，HTML5 に取り入れられ，現在では，Safari，Opera，Firefox のあるバージョン以降，対応している。Internet Explorer（IE）は，当初対応しておらず，IE6 以降，Canvas をエミュレートで対応していたが，最新のブラウザ Microsoft Edge では対応している。

Canvas を JavaScript で使うには，DOM（Document Object Model）によって，canvas 要素を指定し，操作を行う。HTML 文書中の canvas 要素に，

<canvas id=”canvas”></canvas>

というように，”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。

canvas 要素で指定できる属性は，ボックス領域の幅と高さを表す width 属性と height 属性である。

本稿で用いる図形や文字を描く主な Canvas 機能は以下のものである。

strokeRect(x,y,w,h)…(x,y) を左上端とする幅 w，高さ h の四角形を描く

fillRect (x,y,w,h)…(x,y) を左上端とする幅 w，高さ h の塗りつぶし四角形を描く

fillText(t,x,y)…(x,y) から文字データ t を塗りつぶしたテキストを描く

(x, y) から (x’, y’) まで直線を描くには，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。さらに，直線を続けて描いていくと多角形が描け，fill() により塗りつぶすことができる。

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

fillStyle…図形の塗りつぶしの色を指定する

rgba…色を指定する場合に使用し，RGB は赤, 緑, 青を 0～255 の数値で指定する。A は透明度を表し，0（透明）～1（不透明）の値で指定する。

JavaScript は，HTML の中に記述するスクリプト言語で，Java 言語に言語体系が似ているオブジェクト指向言語である。JavaScript は，クライアントコンピュータで動き，どのブラウザも対応している。HTML の <script>～</script> の中に記述する。

JavaScript の中で文字列や計算結果を表示する際には通常 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 で描画する場合に，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いてしまい，そのページには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。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=出力結果;

によって，ID 名 "result" の部分に出力するのである。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 と同じページに出力することができる。

3．Web プログラミングによるグラフ解法のプログラム作成と実行結果

3.1　プログラム作成

ここで扱う線形計画の最大化問題では，変数は 2 つの場合で変数は 2 つとも 0 以上の実数に限定している。また，制約条件式と目的関数の式は傾きが互いに異なるものとしており，最適解が一つの点として求められる場合である。線分上のすべての点といった複数の解がある場合には対応していない。傾きは負の値になることを前提としている。制約条件式の数は 5 つまでとしているが，プログラムを拡張すれば容易に増やすことができる。

制約条件式に x＝定数や y＝定数で表される式が含まれていると x や y の係数が 0 となり，ゼロ除算が生じる為正しく計算ができない。今後，係数が 0 の場合にも対応できるようにプログラムを拡張する必要がある。

プログラム作成の要点については次の通りである。まず，フォームタグを使って，制約条件式と目的関数の式の係数を入力させる。制約条件の各境界直線の傾きを求め，バブルソートアルゴリズムを使って[8]，傾きの絶対値が小さい順に並べ替える。各境界直線の x 切片と y 切片を求め，境界直線同士の交点をすべて求める。第 1 象限から外れている交点は除き，制約条件を満たす領域に含まれない交点も除く。制約条件を満たす領域に含まれるか否かについては，交点と同じ x 座標において，交点の y 座標より下にどれかの境界直線があるかどうかにより判定し，ある場合はその交点は制約条件を満たす領域に含まれないことになる。

こうして，制約条件式を満たす領域を囲む交点だけを抽出し，境界直線が y 軸と最も低いところで交わる点と，境界直線が x 軸と最も原点に近いところで交わる点を求める。これらの点に原点を加えた閉じた図形が制約条件式を満たす領域となる。目的関数（利益）の最大値は，制約条件式を満たす領域の周上の交点をすべて目的関数の式に代入し，目的関数（利益）の値が最大となるときの交点を見出す。これが最適解となる。

その後，Canvas 要素と JavaScript を用いて，グラフを描く。

3.2　実行結果

第 2 章の例題の場合に，図 2 に示されるように，制約条件式の数，制約条件の係数，目的関数の式の係数を入力フォームに入力する。「2 変数線形計画問題のグラフによる解法（最大化問題）」ボタンをクリックすると，プログラムが実行され，実行した結果が図 3 に示されている。

各境界直線が点線で表され，制約条件を満たす領域がグレーで塗りつぶされて示されている。図 1 とスケールは異なっているが，同

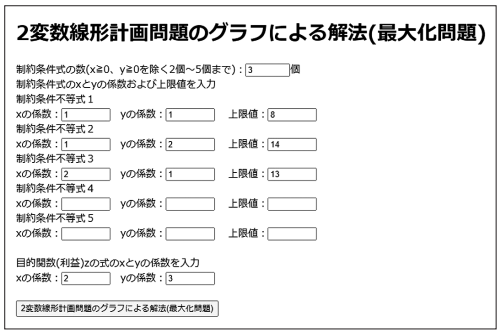

図 2　例題の入力部分
（Microsoft Edge によるブラウザ画面）

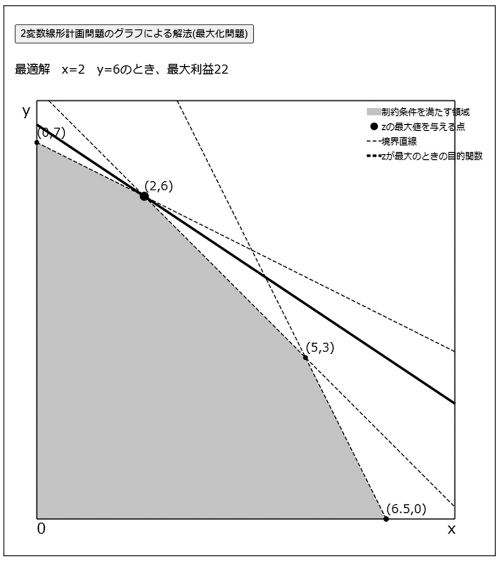

図 3　例題（z＝2x＋3y）の場合の結果
（Microsoft Edge によるブラウザ画面）

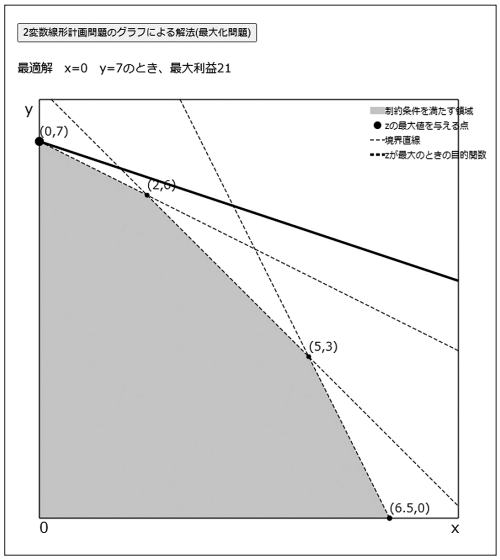

図 4　z＝x＋3y の場合の結果
（Microsoft Edge によるブラウザ画面）

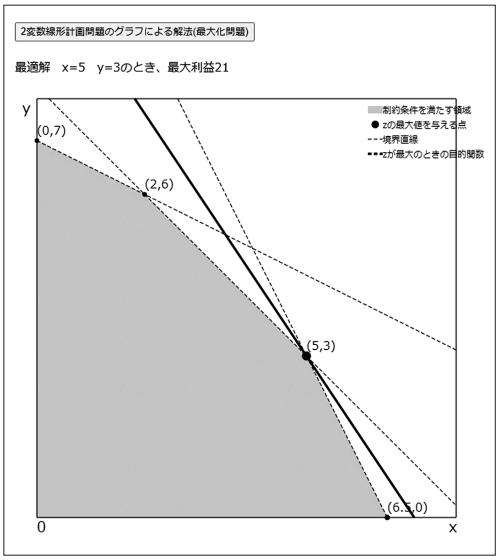

図 5　z＝3x＋2y の場合の結果
（Microsoft Edge によるブラウザ画面）

じ領域を表している。制約条件を満たす領域の周上の交点を小さな黒丸と座標で表し，目的関数（利益）の最大値を与える点を大きな黒丸で示し，最適解を与える x と y およびそのときの最大利益を表示している。太い直線は，利益が最大となるときの目的関数（利益）の直線を表している。

同じ制約条件で目的関数の式が，z＝x＋3y

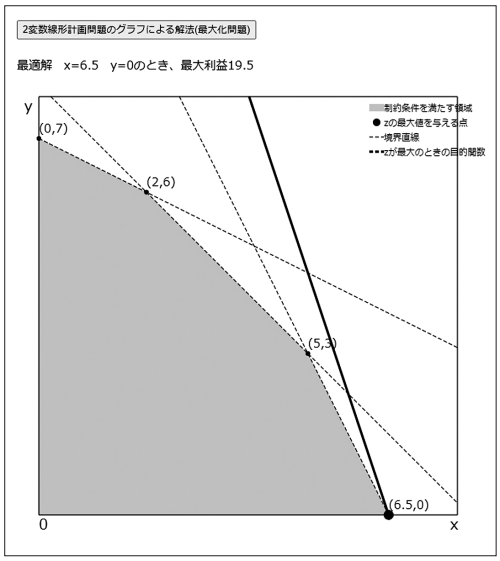

図 6　z＝3x＋y の場合の結果
（Microsoft Edge によるブラウザ画面）

に変わった場合傾きが変わるので，最適解が図 1 の点 A になる。プログラムの実行結果も，図 4 のように点 A に相当する点になっている。

同じ制約条件で目的関数の式が，z＝3x＋2y に変わった場合傾きが変わり，最適解が図 1 の点 C になる。プログラムの実行結果も，図 5 のように点 C に相当する点になっている。

さらに，同じ制約条件で目的関数の式が，z＝3x＋y に変わった場合傾きが変わり，最適解が図 1 の点 D になる。プログラムの実行結果も，図 6 のように点 D に相当する点になっている。

別の例として，制約条件式が 5 つの場合の結果を図 7 に与える。制約条件式が増えても正しく計算されていることがわかる。

図 8 には，このプログラムのソースが表示されている。

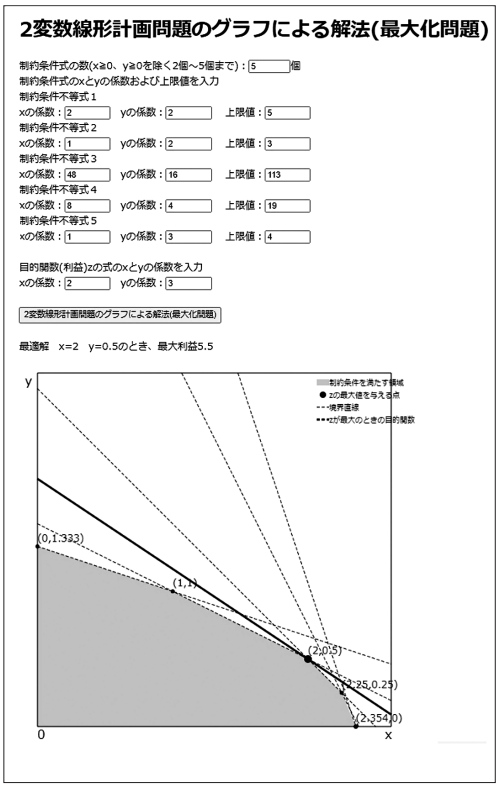

図 7　制約条件が 5 つの場合の結果
（Microsoft Edge によるブラウザ画面）

4．おわりに

本稿では，HTML5 の Canvas と JavaScript を用いて，Web 上で 2 変数線形計画の最大化問題のグラフによる解法を行うプログラムを作成した。プログラムでは，制約条件と目的関数の式を入力すると，制約条件を満たす領域をグラフに描き，目的関数（利益）が最大となる x と y の値と最大値を表示した。

今後は，制約条件式の係数に 0 があるときのようなより一般性を持った場合への拡張や，最小化問題や輸送問題に対応できるようなプログラムを作成していく。

```
<!DOCTYPE HTML>
<html lang="ja">
<head><title>2変数線形計画問題のグラフによる解法(最大化問題)</title></head>
<body>
<h1>2変数線形計画問題のグラフによる解法(最大化問題)</h1>
<form name="form1">
制約条件式の数(x≧0、y≧0を除く2個～6個まで):<input type="text" name="const" size="6">個<br>
制約条件式のxとyの係数および上限値を入力<br>
制約条件不等式1<br>
xの係数:<input type="text" name="x1" size="6">　yの係数:<input type="text" name="y1" size="6">
　上限値:<input type="text" name="u1" size="6"><br>
制約条件不等式2<br>
xの係数:<input type="text" name="x2" size="6">　yの係数:<input type="text" name="y2" size="6">
　上限値:<input type="text" name="u2" size="6"><br>
制約条件不等式3<br>
xの係数:<input type="text" name="x3" size="6">　yの係数:<input type="text" name="y3" size="6">
　上限値:<input type="text" name="u3" size="6"><br>
制約条件不等式4<br>
xの係数:<input type="text" name="x4" size="6">　yの係数:<input type="text" name="y4" size="6">
　上限値:<input type="text" name="u4" size="6"><br>
制約条件不等式5<br>
xの係数:<input type="text" name="x5" size="6">　yの係数:<input type="text" name="y5" size="6">
　上限値:<input type="text" name="u5" size="6"><br>
<br>
目的関数(利益)zのxとyの係数を入力<br>
zの係数:<input type="text" name="z0" size="6">　yの係数:<input type="text" name="y0" size="6"><br>
<br>
<input type="button" value="2変数線形計画問題のグラフによる解法(最大化問題)" onClick="lpmax();"><br><br>
</form>

<div id="result">
</div>
<canvas id="canvas" width="700" height="700"></canvas>
<br><br>

<script>
function lpmax(){
var n,neq,i,j,k,l,a,kmax,lmax,max,isol;

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

//制約条件式の数の上限値
n=5;

//制約条件式の数
neq=Number(document.form1.const.value);

//coef[0～2][i]…i番目の制約条件式のx,yの係数と上限値、coef[0～1][0]…目的関数のx,yの係数
var coef=new Array();
 for(i=0;i<=n;i++){
  coef[i]=new Array();
 }

//フォームの係数を2次元配列coef[][]に代入
coef[0][0]=Number(document.form1.x0.value);
coef[1][0]=Number(document.form1.y0.value);|
coef[0][1]=Number(document.form1.x1.value);
coef[1][1]=Number(document.form1.y1.value);
coef[2][1]=Number(document.form1.u1.value);
coef[0][2]=Number(document.form1.x2.value);
coef[1][2]=Number(document.form1.y2.value);
coef[2][2]=Number(document.form1.u2.value);
coef[0][3]=Number(document.form1.x3.value);
coef[1][3]=Number(document.form1.y3.value);
coef[2][3]=Number(document.form1.u3.value);
coef[0][4]=Number(document.form1.x4.value);
coef[1][4]=Number(document.form1.y4.value);
coef[2][4]=Number(document.form1.u4.value);
coef[0][5]=Number(document.form1.x5.value);
coef[1][5]=Number(document.form1.y5.value);
coef[2][5]=Number(document.form1.u5.value);


//y=ax+bの式に変形し、傾きをcx[]、y切片をcy[]へ代入
var cx=new Array();
var cy=new Array();
for(i=1;i<=neq;i++){
 cx[i]=-Math.round(1000*coef[0][i]/coef[1][i])/1000;
 cy[i]=Math.round(1000*coef[2][i]/coef[1][i])/1000;
}

//傾きの絶対値の小さい順に並べ替えて、cx[i]とcy[i]に格納
for(i=1;i<neq;i++){
  for(j=neq;j>i;j--){
    if(Math.abs(cx[j-1])>Math.abs(cx[j])){
      a=cx[j-1];
      b=cy[j-1];
      cx[j-1]=cx[j];
      cy[j-1]=cy[j];
      cx[j]=a;
      cy[j]=b;
    }
  }
}

//境界直線の交点を求め、座標をpixとpiyに入れる
var pix=new Array();
var piy=new Array();
for(i=0;i<=n;i++){
 pix[i]=new Array();
 piy[i]=new Array();
}
for(i=1;i<neq;i++){
 for(j=i+1;j<neq;j++){
  pix[i][j]=Math.round(1000*(cy[j]-cy[i])/(cx[i]-cx[j]))/1000;
  piy[i][j]=Math.round(1000*(cx[j]*cy[i]-cx[i]*cy[j])/(cx[j]-cx[i]))/1000;
 }
}

//第1象限から外れる交点を除いて、pxとpyに入力
var px=new Array();
var py=new Array();
k=0;
 for(i=1;i<neq;i++){
  for(j=i+1;j<neq;j++){
   if((pix[i][j]>=0)&&(piy[i][j]>=0)){
    k=k+1;
    px[k]=pix[i][j];
    py[k]=piy[i][j];
   }
  }
}
kmax=k;

//制約条件を満たす領域に入っていない交点を除いて、axとayに入れる
var ax=new Array();
var ay=new Array();

l=0;
for(i=1;i<=kmax;i++){
k=0;
 for(j=1;j<=neq;j++){
 a=cx[j]*px[i]+cy[j];
  if(py[i]>a){
    k=k+1;
  }
 }
 if(k==0){
  l=l+1;
  ax[l]=px[i];
  ay[l]=py[i];
 }
}
lmax=l;

//y-切片の最小値を求める
var cyl=new Array();
k=-1;
for(i=1;i<=neq;i++){
 if(cy[i]>=0){
 k=k+1;
 cyl[k]=cy[i];
 }
}
max=k;

a=cyl[0];
for(i=1;i<=max;i++){
 if(a>cyl[i]){
  a=cyl[i];
 }
}
ax[0]=0;
ay[0]=a;

//x-切片の最小値を求める
var cxl=new Array();
k=-1;
for(i=1;i<=neq;i++){
 a=Math.round(1000*coef[2][i]/coef[0][i])/1000;
 if(a>=0){
  k=k+1;
  cxl[k]=a;
 }
}
max=k;

a=cxl[0];
for(i=1;i<=max;i++){
 if(a>cxl[i]){
  a=cxl[i];
 }
}
ax[lmax+1]=a;
ay[lmax+1]=0;


//目的関数(利益)を計算し最大値を求める
var z=new Array();
for(i=0;i<=lmax+1;i++){
 z[i]=coef[0][0]*ax[i]+coef[1][0]*ay[i];
}
isol=0;
for(i=1;i<=lmax+1;i++){
 if(z[i]>z[i-1]){
  isol=i;
 }
}
 result.innerHTML=result.innerHTML+" 最適解　x="+ax[isol]+"　y="+ay[isol]+"のとき、最大利益"+z[isol]+"<br>";

//グラフの作成
var xm,ym,xmax,ymax,xi,yi,xh,yh,xf,yf,x1,y1,x2,y2,h;

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

//(xi,yi)…左上隅の座標、(xf,yf)…右下隅の座標、xh…横軸の幅、yh…縦軸の長さ
 xi=30;
 yi=30;
 xh=570;
 yh=570;
 xf=xi+xh;
 yf=yi+yh;

if(ax[lmax+1]>ay[0]){
 h=Math.floor(1000*0.9*xh/ax[lmax+1])/1000;
}
else{
 h=Math.floor(1000*0.9*yh/ay[0])/1000;
}

//制約条件式を満たす領域の描画
ax[lmax+2]=0;
ay[lmax+2]=0;
cnt.fillStyle="lightgray";
cnt.beginPath();
cnt.moveTo(xi,yf);
for(i=0;i<=lmax+2;i++){
 cnt.lineTo(xi+h*ax[i],yf-h*ay[i]);
}
cnt.closePath();
cnt.fill();

//境界直線の交点とzが最大となる点の描画
cnt.font="10px sans-serif";
cnt.fillStyle="black";
for(i=0;i<=lmax+1;i++){
cnt.beginPath();
if(i==isol){a=8;}
else{a=3;}
x=xi+h*ax[i];
y=yf-h*ay[i];
cnt.arc(x,y,a,0,2*Math.PI,false);
cnt.closePath();
cnt.fill();
cnt.fillText("("+ax[i]+","+ay[i]+")",x,y-8);
}

//グラフ枠の描画
cnt.rect(xi,yi,xh,yh);
cnt.stroke();

//境界線とzが最大のときの目的関数の描画
cx[0]=-Math.round(1000*coef[0][0]/coef[1][0])/1000;
cy[0]=Math.round(1000*z[isol]/coef[1][0])/1000;

for(i=0;i<=neq;i++){
 if(i==0){
  cnt.lineWidth=3;
 }
 else{
  cnt.lineWidth=1;
  cnt.setLineDash([5,3]);
 }
 cnt.beginPath();
 x0=xi;
 y0=yf-h*cy[i];
 if(y0<yi){
  x0=xi+h*(yh/h-cy[i])/cx[i];
  y0=yi;
 }
 cnt.moveTo(x0,y0);
 x1=xi-h*cy[i]/cx[i];
 y1=yf;
 if(x1>xf){
  x1=xf;
  y1=yf-h*(cx[i]*xh/h+cy[i]);
 }
 cnt.lineTo(x1,y1);
 cnt.stroke();
}

//文字の描画
cnt.font="20px sans-serif";
cnt.fillText("x",xf-10,yf+20);
cnt.fillText("y",xi-20,yi+20);
cnt.fillText("0",xi,yf+20);
cnt.fillStyle="lightgray";
cnt.fillRect(xf-120,yi+10,20,10);
cnt.fillStyle="black";
cnt.font="12px sans-serif";
cnt.fillText("制約条件を満たす領域",xf-100,yi+20);
cnt.beginPath();
cnt.arc(xf-110,yi+85,5,0,2*Math.PI,false);
cnt.closePath();
cnt.fill();
cnt.fillText("zの最大値を与える点",xf-100,yi+40);
cnt.beginPath();
cnt.moveTo(xf-120,yi+55);
cnt.lineTo(xf-100,yi+55);
cnt.stroke();
cnt.fillText("境界直線",xf-100,yi+60);
cnt.lineWidth=3;
cnt.beginPath();
cnt.moveTo(xf-120,yi+75);
cnt.lineTo(xf-100,yi+75);
cnt.stroke();
cnt.fillText("zが最大のときの目的関数",xf-100,yi+80);

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

図 8　プログラムソース

参考文献

1）宮川　公男：「経営情報入門」実教出版，1999 年
2）村山　秀明：「HTML5 入門」，工学社，2012 年
3）スタジオ イー・スペース：「HTML5＋CSS 標準テキスト」，技術評論社，2011 年
4）福永　厚：「Web プログラミングによる在庫管理の ABC 分析と PPM」，北海学園大学学園論集第 181 号，pp.33-43，2020 年
5）高橋　麻奈：「やさしい JavaScript の基本」，SB クリエイティブ，2014 年
6）伊藤　静香：「3 日でマスター JavaScript」，ソシム，2014 年
7）河西　朝雄：「ゼロからわかる JavaScript 超入門」，技術評論社，2010 年
8）石田保輝，宮崎修一：「アルゴリズム図鑑」，翔泳社，2017 年