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窓口サービスのデータ解析
著者	福永，厚；FUKUNAGA，Atsushi
引用	北海学園大学学園論集(170)：17-26
発行日	2016-12-25

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

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窓口サービスがある[1)]。窓口サービスの問題は，スーパーマーケットやコンビニエンスストアのレジ，銀行や役所，病院の窓口，駅や会場の切符売り場などのような，サービスを提供する窓口にサービスを受ける為に並ぶ客の列，つまり待ち行列を分析する待ち行列問題でもある。待ち行列問題は，ネットワークのトラフィックや物流など，幅広く様々な問題に適用できる。

窓口サービスの問題では，POS（Point of Sales）システムの導入によるレジのサービス時間の短縮化といったサービスの内容自体には踏み込まず，窓口にできる待ち行列を問題とする。客が窓口に到着する状況から，窓口が一つや二つ或いはそれ以上の場合に，待ち行列の長さや待ち時間，窓口の稼働率などを求めることによって，窓口の数を決める際の意思決定に役立てる。待ち行列が長くできて客が長時間待たされることになると，客の満足度が下がり別の店に行ってしまいかねない一方，窓口を多くして待ち行列を減らし待ち時間を短縮すると，顧客満足度は向上するが窓口の増設や運用にかかるコストが上昇してしまう。従って，窓口の数は，顧客満足度とコストのトレードオフの関係にあり，適正な窓口数を決定することが必要となる。

その為には，実際に客が窓口に到着する時刻やサービス時間についてのデータを収集し，そのデータを用いていろいろな窓口数の場合に，待ち行列や待ち時間，窓口の稼働率といった様々な指標を計算してデータ解析を行い，比較を行わなければならない。収集した多くのデータの解析を行うことは大変な労力であり，技法を習得していない人には難しい作業である為，データを入力するだけで自動的に解析を行えるプログラムが必要とされる。また，単に様々な指標を計算して数値として表すだけでなく，待ち行列の状況をグラフによって可視化し視覚的に表現すると捉えやすい。

筆者は，以前，Microsoft Excel を使って窓口サービスにおける待ち行列のデータ解析を行い，VBA プログラミングによって自動解析や自動グラフ化を行った[2)]。Excel は広く使われているソフトであるが，コンピュータに Excel がインストールされている必要があり，またローカルで動くソフトである。筆者は，さらに Web に対応する為に，プログラミング言語 Java のアプレットを使った場合

の自動化を行った[3]。そこでは，何人かの客のデータを入力すると，客の待ち状況を表すタイムチャートを自動的に作成し，解析結果を提示することができた。Java アプレットは，無料で Web 対応であるが，プラグインが必要であり，セキュリティ上の制限も厳しくなってきている。

本稿では，特別なソフトを必要とせずに，ブラウザで表示できる Web プログラミングによる自動解析と自動グラフ作成を行えるプログラムを作成する。Web プログラミングには JavaScript 言語を使い，自動グラフ作成には HTML バージョン 5（以下 HTML5）の Canvas を用いる。JavaScript と Canvas によって窓口サービスの待ち行列の自動解析とグラフ作成ができれば，Web ブラウザさえあれば他に特別なソフトを要することなく，インターネット上のどこででも Web サービスとして待ち行列解析ができるようになる。

以下，第 2 章では窓口サービスにおける待ち行列解析の概要について，第 3 章では JavaScript と HTML5 の Canvas について，第 4 章では JavaScript と HTML5 の Canvas によるデータ解析およびグラフ作成を行うプログラムについて論述し，第 5 章でまとめる。

2．窓口サービス

窓口サービスで扱う用語に，以下のものがある。

客　：サービスを受けるために窓口に到着する人

窓口：サービスを行う場所

到着：サービスを受ける為にサービス機関に来ること

到着時間間隔：客が到着した時刻から，すぐ前の客が到着した時刻を引いたもの

サービス時間：客がサービスを受けた時間

タイムチャート：縦軸（下方）に客を，横軸に時間を取り，客の待ち状態を黒塗りの四角（■），サービス状態を白塗りの四角（□）で表したもの

待ち行列グラフ：縦軸に待ち行列を，横軸に時間を取り，待ち行列の人数を折れ線グラフで表したもの

データ解析を行って求める諸量は，以下のものである。

全体の終了時間：最初の客が到着してから，最後の客のサービスが終わるまでの時間

最長待ち行列：最も長くできた待ち行列

延べ待ち人数：少しでも待った客の人数

平均待ち時間：客が到着したときに出来ている待ち行列の人数を合計し，客数で割ったもの

最長待ち時間：最も長い待ち時間

合計待ち時間：各客の待ち時間を合計したもの

一人当たりの待ち時間：待ち時間の合計を客数で割ったもの

窓口空き時間：窓口がサービスしていない時間

窓口稼働率＝（全体の終了時間－空き時間）／全体の終了時間

3．HTML5 と JavaScript について

3.1　HTML5 と Canvas

HTML は，HyperText Markup Language の略で，Web ページを記述する言語である。

WWW（World Wide Web）は，スイスのジュネーブにある欧州原子核共同研究機関（CERN）にいた Tim Berners-Lee が，1989 年に研究グループ間でお互いのコンピュータをインターネットでつないで，論文や資料を閲覧できるようにしたことが発端となり，現在では，世界中のコンピュータがインターネットで結ばれ，公開されている情報を Web ページとして見られる仕組みである。公開する Web ページを記述する HTML の言語仕様は，WWW の標準化団体である W3C（World Wide Web Consortium）が管理している。

HTML は，W3C によって現在まで言語仕様が改編されて，1999 年の HTML4.01 になってから 10 年以上続き，新しいバージョンである HTML バージョン 5 が 2014 年に改定された[4),5)]。

HTML5 では，HTML4.01 まで使われていた<font>や ，<frame>などのデザインに関するタグが多く廃止され，<section>や<article>，<header>，<footer>などの文書の構造化をよりよくするためのタグが追加された。

Web ページで図や画像，アニメーションなどの動的コンテンツを描画するためには，Adobe の Flash が代表的であったが，プラグインが必要であり，セキュリティ上の脆弱性などの問題もあってあまり使われなくなって来ている。代わりに 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。ただし，Canvas では，線や四角形を描いたり，色を塗ることはできるが，描いた図を動かくことはできず，アニメーションを作りたい場合は，一コマずつ図を書き直すという処理を行わなければならない。複雑な図形やアニメーションは Flash の方が適しており，Canvas は Flash の代替とは言えない。

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 Internet 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[6)]。HTML 文書中の Canvas 要素に例えば，

<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。さらに，

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，本稿では，領域を広めに取るために，width=”1500” height=”800”で指定している。

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)…座標(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

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。

3.2　JavaScript

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 でグラフを描画する場合に，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 プロパティによって，HTML の内容を書き換える方法を用いる[7),8)]。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，データ解析結果を，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。

4．Web プログラミングによる窓口サービスのデータ解析とタイムチャートの作成

窓口が一つの場合に，客の到着時間間隔とサービス時間のデータを入力すると，自動的にデータ解析を行い，タイムチャートを作成するプログラムを作成した。実行結果として，客が 10 人の場合の入力画面を図 1 に，"窓口サービスのデータ解析（窓口は一つの場合）"ボタンをクリックした場合の結果を図 2 に示す。図 1 においてデータを入力する際に，最初の客の到着時刻を基準 0 としているので，客 1 の到着時間間隔は 0 となっている。図 2 には，データ解析結果とタイムチャートが同じページに描かれている。タイムチャートでは，待ち状態を■，サービス状態を□で表している。また，このプログラムは，人数が 10 人より少ない場合にも適用できる。図 3 には客数が 6 人の場合の入力画面を表示し，図 4 にはボタンをクリックした結果を示している。タイムチャートでは，自動的に最後の客が終わる時刻に合わせたサイズの図になっている。このプログラムでは，客数を 10

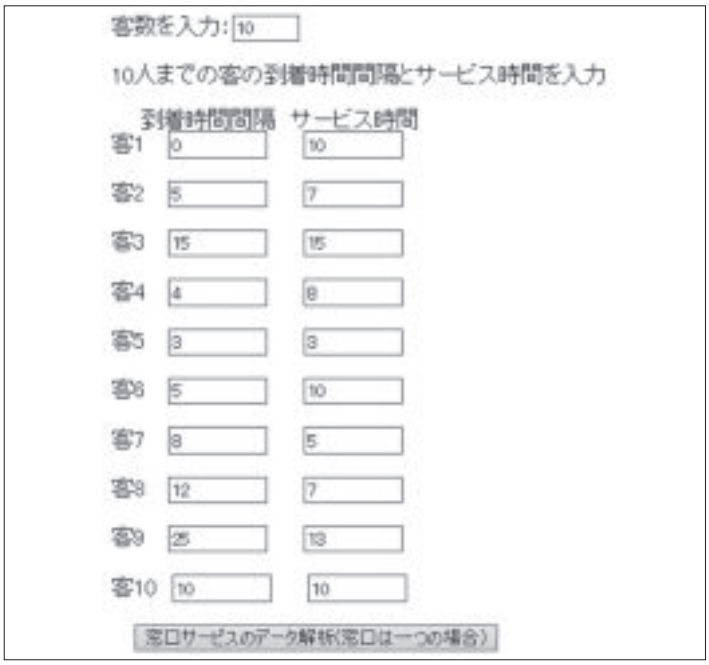

図 1　客が 10 人の場合のデータ入力（Internet Edge によるブラウザ画面）

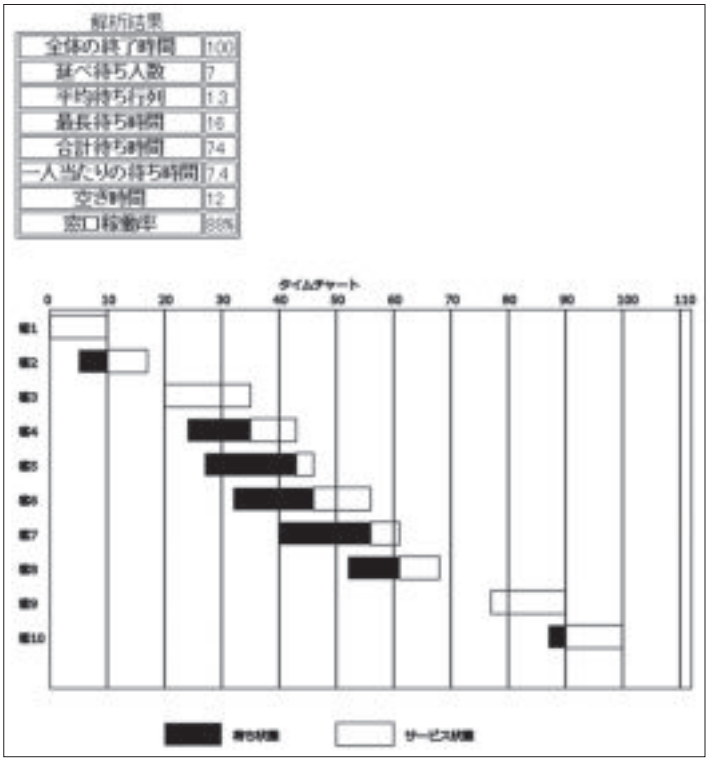

図 2　客が 10 人の場合のデータ解析結果とタイムチャート（Internet Edge によるブラウザ画面）

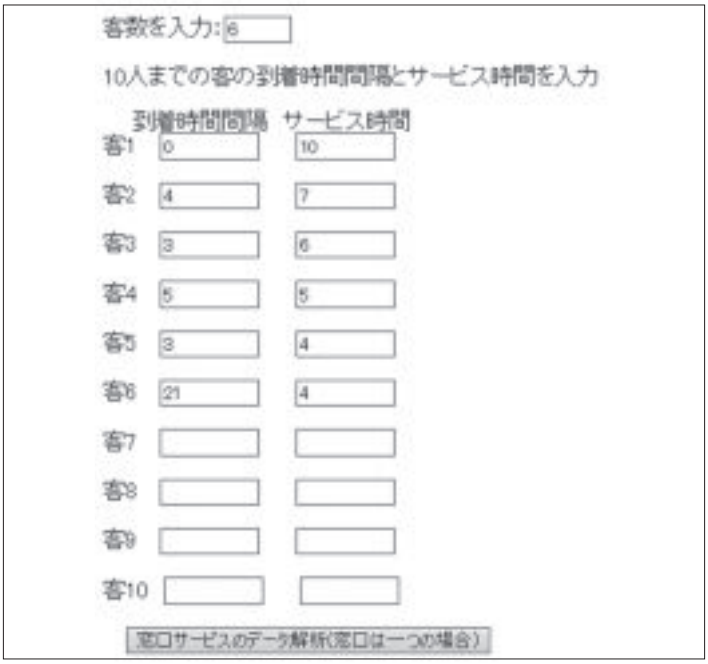

図 3　客が 6 人の場合のデータ入力（Internet Edge によるブラウザ画面）

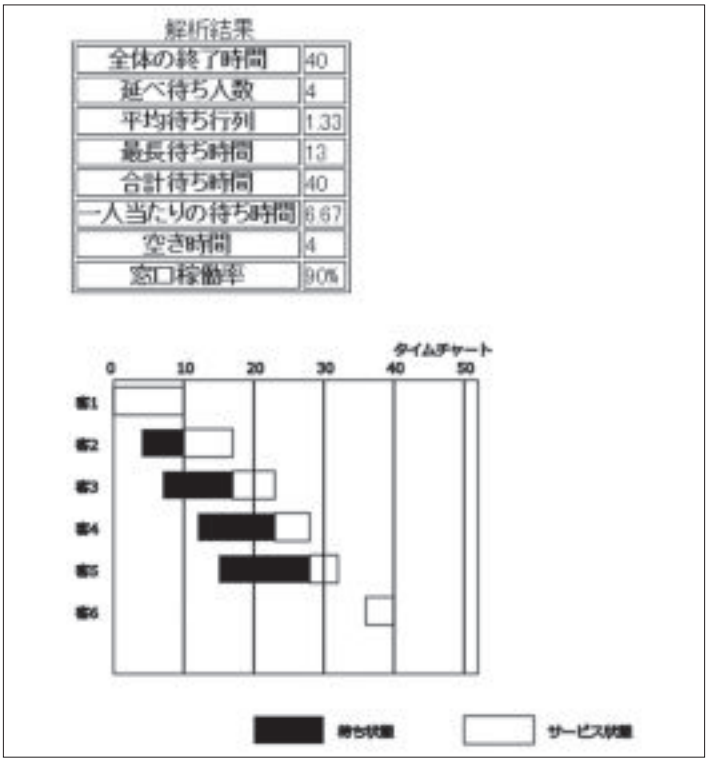

図 4　客が 6 人の場合のデータ解析結果とタイムチャート
（Internet Edge によるブラウザ画面）

人までとしているが，10 人より多い客数に対応するようにプログラムを拡張することは容易である。

図 5 にこのプログラムのソースを表示し，以下でプログラムの内容について要約する。

Web 上での入力には，HTML のフォームタグを用いている[9]。客数とそれぞれの客の到着時間間隔，サービス時間のデータの入力に，HTML のフォームタグのテキストボックス

<input type=”text” name=”○” size=”□”>

を用いている。size 属性は，表示する半角文字数を指定する。

<input type=”button” value=”窓口サービスのデータ解析（窓口は一つの場合）” onClick=”queuing();”>

という文は，イベントボタンを表し，このボタンをクリックした場合に，JavaScript 関数”queuing()”を実行する。

JavaScript 関数 queuing()では，まず，2 次元配列変数 kyaku を宣言している。JavaScript の 2 次元配列宣言では，先に kyaku[]という行要素の宣言を行い，次に各行要素ごとに列要素を宣言することによって 2 次元配列を実現している。2 次元配列 kyaku[i][j]では，行要素 i は 0 〜n−1 の値を取

```
<!DOCTYPE HTML>
<html lang="ja">
<head><title>窓口サービスのデータ解析</title></head>
<body>
<p>
<form name="form1">
客数を入力：<input type="text" name="kyakusuu" size="6"><br>
<p>10人までの客の到着時間間隔とサービス時間を入力</p>
　　到着時間間隔　サービス時間<br>
客1　<input type="text" name="arrival0" size="10">
　<input type="text" name="service0" size="10"><br><br>
客2　<input type="text" name="arrival1" size="10">
　<input type="text" name="service1" size="10"><br><br>
客3　<input type="text" name="arrival2" size="10">
　<input type="text" name="service2" size="10"><br><br>
客4　<input type="text" name="arrival3" size="10">
　<input type="text" name="service3" size="10"><br><br>
客5　<input type="text" name="arrival4" size="10">
　<input type="text" name="service4" size="10"><br><br>
客6　<input type="text" name="arrival5" size="10">
　<input type="text" name="service5" size="10"><br><br>
客7　<input type="text" name="arrival6" size="10">
　<input type="text" name="service6" size="10"><br><br>
客8　<input type="text" name="arrival7" size="10">
　<input type="text" name="service7" size="10"><br><br>
客9　<input type="text" name="arrival8" size="10">
　<input type="text" name="service8" size="10"><br><br>
客10　<input type="text" name="arrival9" size="10">
　<input type="text" name="service9" size="10"><br><br>

<input type="button" value="窓口サービスのデータ解析(窓口は一つの場合)"
 onClick="queuing();"><br><br>
</form>
<div id="result">
</div>
<canvas id="canvas" width="1500" height="800"></canvas>
<br><br>
</p>

<script>
function queuing(){
var i,j;
var kyaku=new Array(10);
 for(i=0;i<10;i++){
  kyaku[i]=new Array(6);
 }
n=Number(document.form1.kyakusuu.value);
kyaku[0][0]=Number(document.form1.arrival0.value);
kyaku[1][0]=Number(document.form1.arrival1.value);
kyaku[2][0]=Number(document.form1.arrival2.value);
kyaku[3][0]=Number(document.form1.arrival3.value);
kyaku[4][0]=Number(document.form1.arrival4.value);
kyaku[5][0]=Number(document.form1.arrival5.value);
kyaku[6][0]=Number(document.form1.arrival6.value);
kyaku[7][0]=Number(document.form1.arrival7.value);
kyaku[8][0]=Number(document.form1.arrival8.value);
kyaku[9][0]=Number(document.form1.arrival9.value);

kyaku[0][1]=Number(document.form1.service0.value);
kyaku[1][1]=Number(document.form1.service1.value);
kyaku[2][1]=Number(document.form1.service2.value);
kyaku[3][1]=Number(document.form1.service3.value);
kyaku[4][1]=Number(document.form1.service4.value);
kyaku[5][1]=Number(document.form1.service5.value);
kyaku[6][1]=Number(document.form1.service6.value);
kyaku[7][1]=Number(document.form1.service7.value);
kyaku[8][1]=Number(document.form1.service8.value);
kyaku[9][1]=Number(document.form1.service9.value);

//絶対到着時刻の計算
kyaku[0][2]=kyaku[0][0];
for(i=1;i<n;i++){
 kyaku[i][2]=kyaku[i-1][2]+kyaku[i][0];
}

//待ち状態開始時刻の初期値=絶対到着時刻
for(i=0;i<n;i++){
 kyaku[i][3]=kyaku[i][2];
}

//最初の客のサービス開始時刻と終了時刻
kyaku[0][4]=kyaku[0][2];
kyaku[0][5]=kyaku[0][4]+kyaku[0][1];

//2番目以降の客のサービス開始時刻と終了時刻
for(i=1;i<n;i++){
 if(kyaku[i][2]>=kyaku[i-1][5]){
  kyaku[i][4]=kyaku[i][2];
  kyaku[i][5]=kyaku[i][4]+kyaku[i][1];
 }
 else{
  kyaku[i][3]=kyaku[i][2];
  kyaku[i][4]=kyaku[i-1][5];
  kyaku[i][5]=kyaku[i][4]+kyaku[i][1];
 }
}
```

図 5　プログラムソース

```
//延べ待ち人数の計算
var nobe=0;
for(i=1;i<n;i++){
 if(kyaku[i][4]>kyaku[i][3]){
        nobe=nobe+1;
 }
}

//平均待ち行列の計算
var avemati,nobe2=0;
for(i=1;i<n;i++){
 if(kyaku[i][4]>kyaku[i][3]){
  nobe2=nobe2+1;
        for(j=0;j<i;j++){
        if((kyaku[i][3]>=kyaku[j][3])&&(kyaku[i][3]<kyaku[j][4])){
        nobe2=nobe2+1;
        }
        }
 }
}
avemati=Math.round(nobe2/n*100)/100;

//最長待ち時間、待ち時間の合計、空き時間、稼働率の計算
var matijikan=0,akijikan=0,matihitori,kadou,maxmati=0;
for(i=1;i<n;i++){
 if(kyaku[i][4]-kyaku[i][3]>0){
  if((kyaku[i][4]-kyaku[i][3])>maxmati){
        maxmati=kyaku[i][4]-kyaku[i][3];
  }
  matijikan=matijikan+kyaku[i][4]-kyaku[i][3];
 }
 if(kyaku[i][2]-kyaku[i-1][5]>0){
  akijikan=akijikan+kyaku[i][2]-kyaku[i-1][5];
 }
}
matihitori=Math.round(matijikan/n*100)/100;
kadou=Math.round(1000*(kyaku[n-1][5]-akijikan)/kyaku[n-1][5])/10;

//解析結果の出力
var result=document.getElementById("result");
result.innerHTML="<table border><caption>解析結果</caption>"
+"<tr><th>全体の終了時間</th><td>"+kyaku[n-1][5]
+"</td></tr><tr><th>延べ待ち人数</th><td>"+nobe
+"</td></tr><tr><th>平均待ち行列</th><td>"+avemati
+"</td></tr><tr><th>最長待ち時間</th><td>"+maxmati
+"</td></tr><tr><th>合計待ち時間</th><td>"+matijikan
+"</td></tr><tr><th>一人当たりの待ち時間</th><td>"+matihitori
+"</td></tr><tr><th>空き時間</th><td>"+akijikan
+"</td></tr><tr><th>窓口稼働率</th><td>"+kadou
+"%</td></tr></table><br><br>";

//タイムチャートの作成
var c=document.getElementById("canvas");
var cnt=c.getContext("2d");
f=5;
a=30;
b=30;
cnt.strokeRect(a,b,a+kyaku[n-1][5]*f+30,30*n+30);
var hmax;
hmax=Math.floor(kyaku[n-1][5]/10)+1;
for(i=1;i<=hmax;i++){
 cnt.beginPath();
 cnt.moveTo(a+f*i*10,b);
 cnt.lineTo(a+f*i*10,b+30*n+30);
 cnt.closePath();
 cnt.stroke();
}
for(i=0;i<n;i++){
 xini=kyaku[i][4]*f+a;
 w=kyaku[i][1]*f;
 yini=b+5+30*i;
 cnt.strokeRect(xini,yini,w,20);
 if(kyaku[i][3]!=kyaku[i][4]){
        xini=kyaku[i][3]*f+a;
        w=(kyaku[i][4]-kyaku[i][3])*f;
        cnt.fillRect(xini,yini,w,20);
 }
}

for(i=1;i<=n;i++){
 cnt.strokeText("客"+i,a-27,b-10+30*i);
}
for(i=0;i<=hmax;i++){
 cnt.strokeText(""+i*10,a-5+f*i*10,b-5);
}
cnt.strokeText("タイムチャート",a+200,b-18);
cnt.fillRect(a+100,b+30*n+60,50,20);
cnt.strokeText("待ち状態",a+150+10,b+30*n+75);
cnt.strokeRect(a+100+150,b+30*n+60,50,20);
cnt.strokeText("サービス状態",a+300+10,b+30*n+75);
}
</script>
</body>
</html>
```

図 5　続き

り，i=0 が客 1 に対応し，それぞれ 10 人までの客に対応している。列要素 j は，0…到着時間間隔，1…サービス時間，2…絶対到着時刻（客 1 が到着してからの時刻），3…待ち状態開始時刻の初期値，4…サービス開始時刻，5…サービス終了時刻を表している。

フォーム（name 属性を”form1”）に入力された客数や到着時間間隔，サービス時間を JavaScipt の変数に代入するには，

Number（document．フォーム名．テキストボックス名 .value）

のように，フォームのテキストボックスに入力された値（value）を Number によって数値に変換している。

図 5 の JavaScript プログラムの待ち行列データ解析部分の要約を，以下で行う。

2 番目以降の客の待ち状態開始時刻の初期値を絶対到着時刻に設定し，その時刻に前の客のサービスが終わっていれば，待ち状態開始時刻＝サービス開始時刻とし，そうでなければ，前の客のサービス終了時刻がこの客のサービス開始時刻となり，サービス開始時刻にサービス時間を加えた時刻がサービス終了時刻となる。

これによって，それぞれの客について，待ち状態開始時刻，サービス開始時刻，サービス終了時刻が得られる。最後の客のサービス終了時刻が全体の終了時間となる。

延べ待ち人数は，待ち時間が 0 より大きい客の人数であるので，

サービス開始時刻＞待ち状態開始時刻

である客の数を計算している。

平均待ち行列は，それぞれの客が到着した時刻にできた待ち行列の人数の平均値であるので，客が到着したときに自分を含めてできた行列の人数をすべての客について合計し，客数で割って求めている。

待ち時間は，

待ち時間＝サービス開始時刻－待ち状態開始時刻

となっており，それらを使って，最長待ち時間，待ち時間の合計，合計を客数で割り一人当たりの待ち時間を計算している。窓口の空き時間の合計は，

次の客の絶対到着時刻－客のサービス終了時刻

をすべての客について合計して求められる。

窓口稼働率は，

窓口稼働率＝（全体の終了時間－空き時間の合計）／全体の終了時間

で計算される。

これらの計算結果は，innerHTML プロパティを用いて，HTML 内の ID”result”領域に，

result.innerHTML=出力結果；

によって出力される。

タイムチャートは，ID 名”canvas”領域に表示する。客数と最後の客のサービス終了時刻に合わせて，strokeRect メンバーを使って図の外枠を描き，beginPath，moveTo，lineTo，closePath，stroke によって縦軸の 10 刻みの目盛り線，strokeText によって縦軸と横軸のラベルを描いている。客の待ち状態を fillRect によって■，サービス状態を strokeRect によって□を描いている。

5．おわりに

本稿では，JavaScript と HTML5 の Canvas を使って，窓口が一つの場合に，Web 上で客の到着時間間隔とサービス時間のデータを入力すると，自動的に待ち行列に関する諸量についてデータ解析を行って計算結果を表示し，さらにタイムチャートを描くプログラムを作成した。

HTML5 にはじめて採用された Canvas は，JavaScript と連携して Web ページ上にグラフィック機能を提供することができ，これを用いると，窓口サービスのタイムチャートが作成できることが示された。

JavaScript の innerHTML プロパティを使うと，データを入力するページと同じページに，計算結果とタイムチャートを描くことができた。

今後は，待ち行列グラフの作成や，窓口が２つ以上の場合への対応，シミュレーションを行う。

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 VBA プログ

ラミングによる待ち行列情報のデータ解析」，北海学園大学学園論集 110 号，pp.79–92，2001 年
3）福永　厚：「経営科学のための Java プログラミングによる待ち行列のデータ解析」，北海学園大学経済論集第 49 巻第 4 号，pp.149–157，2002 年
4）村山　秀明：「HTML5 入門」，工学社，2012 年
5）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
6）高橋　麻奈：「やさしい JavaScript の基本」，SB クリエイティブ，2014 年
7）伊藤　静香：「3 日でマスター JavaScript」，ソシム，2014 年
8）河西　朝雄：「ゼロからわかる JavaScript 超入門」，技術評論社，2010 年
9）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