

2022年度 卒業論文

待ち行列問題シミュレータ教材の開発

指導教員 須田 宇宙 准教授

千葉工業大学 情報ネットワーク学科
須田研究室

1932008 氏名 伊豆原 嵩章

提出日 2023年1月17日

目次

1	はじめに	1
2	乱数	2
2.1	一様乱数	2
2.2	正規乱数	4
2.3	指数乱数	5
3	待ち行列問題	6
3.1	待ち行列問題	6
3.2	モンテカルロ法	8
3.3	安定状態	8
4	開発に使用した言語について	9
4.1	HTML	9
4.2	CSS	11
4.3	JavaScript	13
5	開発したシミュレータについて	14
5.1	教材に必要な機能	14
5.2	開発した教材の実行例	16
5.3	開発したプログラムについて	19
6	おわりに	26
7	謝辞	27

1 はじめに

病院の受付、店舗や交通渋滞など我々の生活の至る所に待ち行列が存在する。行列による問題は多岐にわたり生産コストや回転率など、店舗の経営などに大きく関わってくる。最大限の利益を得るためにも、商品やサービスを提供するお客さんの回転率を高めることが重要である。回転率を上げることは、お客さんの待ち時間が少なくなり、提供するサービスの向上にもつながる。それら問題のアプローチ法としてモンテカルロ法を用いた待ち行列問題が存在し、シミュレーションを行うことでリソースを適正に算出し配備することができる。待ち行列問題を用いることで、窓口数や作業員などに対応した客の平均的な待ち時間などを計算することができる。そのため、店舗などではなるべくお客さんの待ち時間を減らし、客の回転率を上げるとともに無駄な設備投資を避けるべく、世の中の様々なところで応用されている。例としてスーパーのレジの数、銀行の窓口数やコールセンターのオペレーターの人数などが挙げられる [2]。

また数値計算の授業において実際に教科書や黒板からの情報だけでは直感的に理解することが困難である。そしてモンテカルロ法的一种である「待ち行列問題」があるが、数式にパラメータを代入して答えを算出する問題と違い、教科書のような静的な教材のみでは理解しづらい。実際に解こうとしても高い精度の解を出すために数万回程度の試行が必要となり手計算での確認は困難である。そして先行研究として視覚的・直感的に理解できることを目的とした待ち行列問題シミュレータが存在する [1]。しかし開発環境が古く現在は使用できないこと、アニメーションが画像を切り替えていくことによる擬似的なアニメーションであり、なめらかなアニメーションではなかった。アニメーションが擬似的なものであり動きが少なく現実の動きと相違があることが問題であると考えた。

そこでアニメーションをより現実的な動きに寄せブラウザ上への移植及び改良することでより利用者の理解度の向上が見込めると考えた。本研究では待ち行列についてアニメーションを用いたシミュレータ教材をブラウザ上で開発することを目的とする。

2 乱数

乱数とは数の出方が不規則で予測がつかず、次に何が出るか分からない数の数列である。乱数は簡単に作ることができ、そして身の周りの生活にも多くの乱数が存在する。ゲームの開発や宝くじの当選番号などもそのうちの1つである。また多くのプログラミング言語に乱数を生成するメソッドや関数が用意されており、使用するのは容易である。乱数は自然現象をシミュレーションするために用いられることが多い。待ち行列問題もその自然現象のうちの1つである。待ち行列問題も乱数を用いてシミュレーションできるが、この現象を再現するためには大量の乱数が必要となる。そのため、コンピュータを用いて乱数を発生させるのが一般的である。しかし、コンピュータは規則に従って動作しているので、完璧な乱数を作ることができないが、それに近い乱数を発生させることは可能である。このようにコンピュータを用いて発生させた乱数のことを擬似乱数と呼ぶ。モンテカルロ法では以下に示すような3つの乱数を用いて計算を行う。

2.1 一様乱数

乱数のうち、出た数が図1のような一様分布に従う乱数を一様乱数という。 x 軸は取り得る値を表し、 y 軸は確率の頻度・密度を表している。図1を見ての通り、どの値でも一定の値をとる。つまり、ある区間において数値が同じ確率で出現する確率分布に従う乱数のことである。一様乱数として身近なもので、サイコロがある。サイコロは1から6までの数値がすべて $\frac{1}{6}$ の確率で出現する。

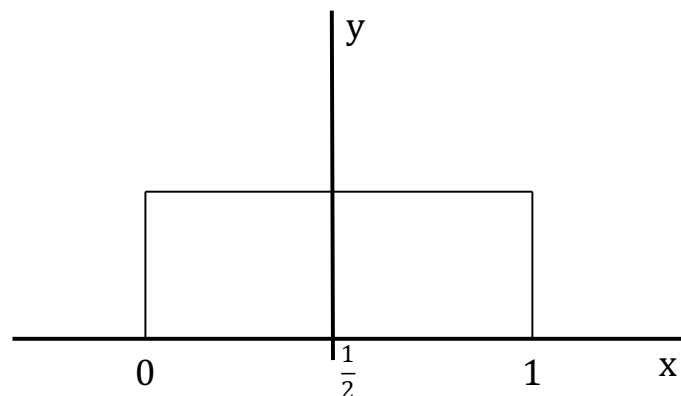


図1 区間(0,1)の一様乱数の確率密度関数

一様乱数の発生方法として、次の2通りが存在する。1つ目はフォン・ノイマン (John Von Neumann) によって提案された平方採中法である。図2は、平方採中法の方法を順を追って説明している。この方法は例えば4桁の乱数列を求める場合、図2に示すように、まず4桁の初期整数1234を平方して、その解の中央の4桁を取り出す。その乱数列5227を第2項とし、以下同様の方法で乱数列を求めていく方法である。

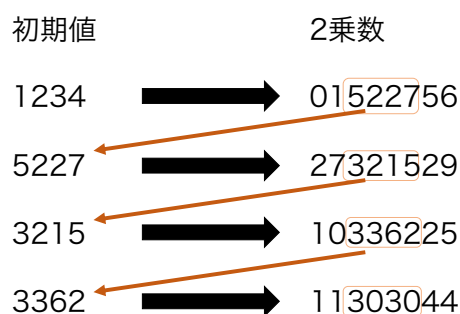


図2 平方採中法による乱数生成の例

平方採中法は計算で必要とする整数の有効桁数が求めたい整数の2倍必要になる上に、乱数列を得るための計算時間も長くなってしまふ。

そこで2つ目の方法として、レーマー (H. Lehmer) により提案された乗算合同法がよくつかわれている。乗算合同法は除算の剰余を数列化していくものである。次式を使い、乱数を求める方法である。

$$x_{n+1} = ax_n \pmod{M} \quad (\text{A})$$

$$n \geq 1 \quad (\text{B})$$

また区間(0,1)の範囲で一様乱数を必要とする場合には、上記の方法で得られた乱数列をMで除算すればよい。乗算合同法では、乱数を生成する方法として剰余を用いているので、乱数の最大値はMより大きくなることはない。必ずMより小さな周期を持つことになる。そのためMとしては大きな値を与える必要がある。しかし式Aから分かるように乱数生成の計算に必要な整数の最大値(aM)になるので、この値がコンピュータの扱える最大整数値を超えないようにする必要がある。

2.2 正規乱数

図3は正規分布の密度関数を表している。x軸は取り得る値を表し、y軸は確率の頻度・密度を表している。事象として平均値を取る確率が一番大きく、平均値から離れるにつれその値を取る確率は小さくなるのがグラフから読み取れる。乱数のうち、生成の割合が図3のような正規分布に依存しているものを正規乱数という。

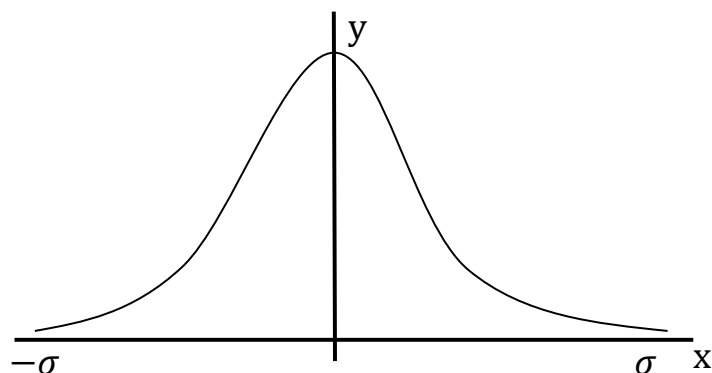


図3 (μ, σ) の正規分布の密度関数

正規乱数は「ある値を中心にして、偶発的に生じる事柄は正規分布に従う」という法則に基づいて生成することができる。また正規乱数は一様乱数から生成することができる。区間 $(0,1)$ で一様乱数を取りだして平均し、これを繰り返して新たな乱数の集合を作る。一様な乱数を取り出してこの平均を取るのので、その値は区間の平均値 $(\frac{1}{2})$ を中心にばらつく。またこのようにして生成した乱数列の分布は、平均値が $\frac{1}{2}$ 、標準偏差が $\frac{1}{\sqrt{12}}$ という正規分布になるという性質がある。このことを中心極限定理と呼び、これを利用して正規乱数に極めて近い分布を持つ乱数列を生成することができる。

取り出す乱数の個数 N はある程度大きな数字が必要であるが、 N を 12 とすれば平均値も標準偏差も整数で表せられるので、一般的には $N = 12$ が使われている。

取り出した乱数の総和 $(v = \gamma_1 + \gamma_2 + \cdots + \gamma_{12})$ の集合は全て 12 倍されて、平均値は 6、標準偏差が 1 になるので、これらの集合から 6 引いた値

$$v = v = \gamma_1 + \gamma_2 + \cdots + \gamma_{12} - 6 \quad (\text{a})$$

は平均値が 0 で、標準偏差が 1 の正規乱数となる。

よって平均値が μ で、標準偏差が σ の正規乱数は

$$\mu + \sigma v \quad (\text{b})$$

という関数で表すことができる。

2.3 指数乱数

図4は指数分布のグラフである。x軸は取り得る値を表し、y軸は確率の頻度・密度を表している。指数乱数はランダムなイベントの発生間隔を表す分布である。日常で使用されている例として「地震が起きる間隔」や「電球の寿命」などを表す分布として使われている [7]。乱数のうち、生成の割合が図4のような指数分布に依存する。

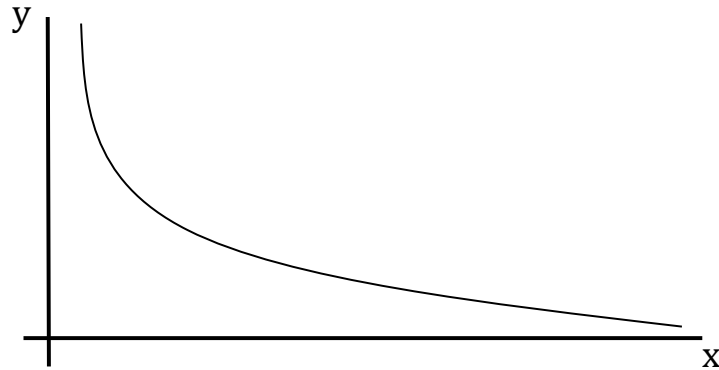


図4 指数分布の概形

性質として「小さな値は多く生じ、大きな値はあまり起こらない」偶発的な値を発生させるということがある。指数乱数の生成方法は、ある分布の乱数の確率分布を積分すると、一様乱数になるという性質を利用して生成する。このような生成方法を逆変換法という。例えば v を指数乱数とし、 γ を一様乱数とすると次のような関係式になる。

$$\int_0^\gamma \alpha \exp(-\alpha v) dx = \gamma \quad (\text{i})$$

この式を積分して変形すると

$$v = -\left(\frac{1}{\alpha}\right) \ln(1 - \gamma) \quad (\text{ii})$$

が得られる。ここで γ は区間 (0,1) の一様乱数なので、 $1-\gamma$ も一様乱数になる。

よって指数乱数の生成方法は

$$v = -\left(\frac{1}{\alpha}\right) \ln \gamma \quad (\text{iii})$$

という式に基づく。この指数分布による乱数は、ある系に偶発的に顧客が入ってくる時間間隔のモデルとしてよく用いられ、式 iii で表された客の流れを最も簡単な流れ、またはポアソン分布と呼んでいる。この場合、パラメータ α は客の流れ密度に相当する。

3 待ち行列問題

3.1 待ち行列問題

待ち行列問題とは、 n 箇所の窓口が開いていてサービスに δ 時間かかる時の窓口で客がサービスで受けられるまでの平均待ち時間 t を計算することである。このとき $(N+1)$ 番目の客が入ってくる時刻 t_{N+1} は下記の式で表すことができる。

$$t_{N+1} = t_N + \tau \quad (1)$$

$$t_0 = 0 \quad (2)$$

$$\tau = -\frac{1}{\alpha} \ln \gamma \quad (3)$$

同様に、個々の客のサービス時間は以下の式で表せれる。

$$\delta = \delta_0 + \sigma v \quad (4)$$

(τ ：次の客が入ってくる時間間隔の期待値， α ：客の流れの密度， γ ：区間 $(0,1)$ の一様乱数， δ ：客一人が受けるサービスの時間， δ_0 ：サービスに必要な平均時間， σ ：標準偏差， $v = \gamma_1 + \gamma_2 + \dots + \gamma_{12} - 6$) 実際に待ち行列問題を解くためには以下の 4 つのパラメータを定め、乱数を発生させて計算していく。

- (I) 客の流れ密度 α
- (II) 窓口の数 n
- (III) サービスに必要な平均時間 δ_0
- (IV) サービスに必要な時間の標準偏差 σ

図5は窓口サービスと顧客入場の関係について図で表したものである． t_k は待ち行列を表しており，窓口である n_m が客で埋まっている時， t_k に格納される．客の入場頻度は指数乱数に基づき，処理時間は正規乱数に基づいている． N 番目の客はどこかの窓口が空いていればその場所でサービスを受けられるが，全てが使用中だといずれかの窓口が空くまで待たされる．この待ち時間を集積し全客数で割れば一人当たりの平均待ち時間が計算される．しかし実際には営業時間や曜日等によって客の入ってる分布が異なるため，乱数などの偶発的な確率変数を用いて試行錯誤的に問題を解いていく数値計算法である．

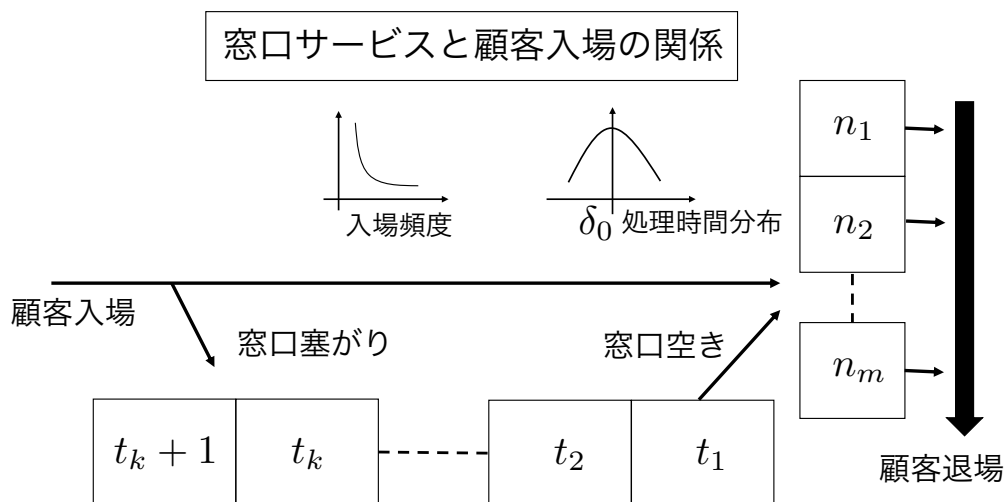


図5 窓口サービスと顧客入場の関係

3.2 モンテカルロ法

モンテカルロ法とは、乱数を用いたシミュレーションを行うことにより近似解を求める計算方法である。この方法を使うことにより解析的に解くことができない問題でも、多くのシミュレーションを繰り返すことによって近似解を求めることができる。応用範囲が広く、問題によっては他の数値計算法より簡単に適用できるが、問題として高い精度を得ようとすれば計算回数が膨大になってしまうという欠点もある。例として、精度を 1 桁上げるためには試行回数 N を 100 倍にしなければならない。しかし、この欠点も近年ではコンピュータの計算能力の向上によって解消されつつある。

3.3 安定状態

安定状態とは、シミュレーションを行なった際、顧客の平均待ち時間が一定で安定している状態のことをいう。この時の平均待ち時間は顧客 1 人 1 人の待ち時間を足していき顧客の数で割って求める。また安定状態にある時の条件は、窓口での処理時間 x 、流れ密度 α 、窓口の数 y の関係が

$$x\alpha < y \quad (\text{I})$$

が成り立つ時に安定状態となる。

4 開発に使用した言語について

4.1 HTML

HTMLとは「Hyper Text Markup Language (ハイパーテキスト・マークアップ・ランゲージ)」のことで、主に Web ページを作成するための言語である [3]。現在のインターネットで公開されている Web ページのほとんどは、HTML で作成されている。ハイパーテキストに目印をつける言語と日本語に訳することができる。

また Hyper Text (ハイパーテキスト) とは、ハイパーリンクを埋め込むことができる高機能なテキストのことであり、文書中の文字や画像に Web サイトの URL を設定することが可能である。そして HTML の特徴として、このハイパーリンク機能で関連する情報同士を結びつけて、情報を整理するということが挙げられる。ハイパーリンクの概念図を図 6 に示す。

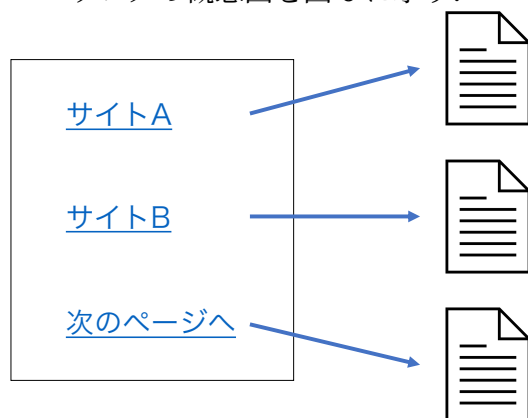


図 6 ハイパーリンクとは

またもう 1 つの特徴として、見出し・段落・表など HTML 文書内で各部分の役割が分かるように目印をつけることがある。これら各部分を element (要素) と呼ぶ。HTML の要素について、図 7 に示す。

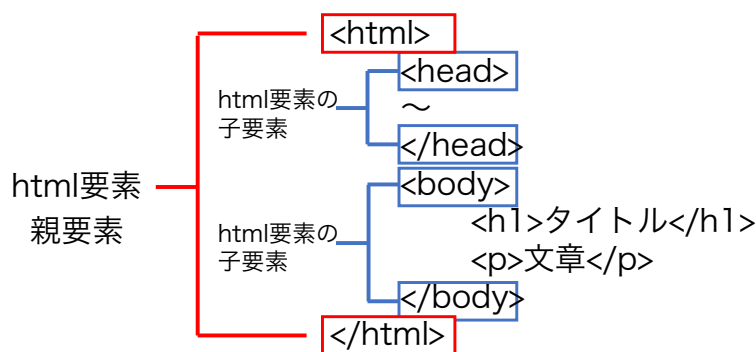


図 7 HTML の要素とは

そして、これら **element** を **HTML** 内に定義するための記号として **HTML タグ** というものが存在する。代表的なものとして以下のものがある。

- (i) `<html></html>`: HTML 文書を定義、コード全体を囲む
- (ii) `<body></body>`: 本文の設計を定義、WEB ページに実際に表示される部分
- (iii) `<a>`: ハイパーリンクを定義
- (iv) `<title</title>`: タイトルを定義

HTML は基本的にこれらのタグを使用して要素を囲んでいくことで作成することができるため、プログラミング学習初心者にも分かりやすい言語とされている。

4.2 CSS

CSS とは「Cascading Style Sheets (カスケーディング・スタイル・シート)」のことであり、WEB ページのスタイル・デザインを指定するための言語である [4]。基本的に HTML と組み合わせて使用する。HTML で定義した見出しや文章の色や大きさ、要素の位置関係など細かく指定することができる。HTML 単体で Web ページを作成することができるが、CSS と組み合わせることで様々な変化をもたらすことができるので、CSS は Web ページのデザイン性を高める上で有効である。

また CSS のスタイル定義の仕組みとして、「セレクトタ」「プロパティ」「値」の 3 要素が存在する。これらの要素は CSS の「どこの」、「何を」、「どうするのか」を指定している。もう少し具体的に説明すると「セレクトタ」はスタイル適用する場所を示し、「プロパティ」はどんな属性を変えるのかを示し、「値」では指定された場所にある属性を「どう」変えるのかを指定している。実例として図 8 に示す。

セレクトタ{プロパティ : 値;}

title{color : yellow;}

タイトルタグの{色を : 黄色にする;}

図 8 CSS の使い方例

4.2.1 CSS の必要性

Web ページを作成する上で HTML だけで見栄えを制御することもできなくはない。しかし、Web ページのスタイリングに HTML を用いるべきではない。理由として HTML は情報構造を定義するための言語であり、スタイリングのような本来の役割とは違った使い方をするとコンピュータや検索エンジンに理解されないでたらめな文書構造になってしまうことがある。ユーザーの検索エンジンによってうまく Web ページが表示されなくなってしまう。また CSS ファイルとして独立させることで、HTML 内にある関連箇所を逐一直さなくても、CSS ファイルを変更するだけで一括で変更することが可能である。つまりメンテナンスが行いやすくなるということも必要性として挙げられる [5]。

4.3 JavaScript

JavaScript とは Web サイトやシステム開発に使われている言語である [6]。動的な Web ページを作成するが得意でユーザーのアクションに応じたコンテンツの表示や、グラフィックアニメーションなども表示することができる。HTML と CSS で構築された Web サイトに動きを加えたり、様々な Web サービスを実現できる。つまり、HTML と CSS に対して指示出しの役割を持つプログラミング言語である。

4.3.1 Java との違い

JavaScript と Java, 名前からして似ているプログラミング言語かと思う。いくつかの共通点はあるが、全く別の言語である。「JavaScript」は開発当初「LiveScript」と呼ばれていたが、その当時人気のあったプログラミング言語の「Java」にあやかって「JavaScript」という名前に変更された。違いとして、「Java」はオブジェクト指向プログラミング言語であるのに対して、「JavaScript」はオブジェクト指向スクリプト言語である。スクリプト言語とは、プログラミングの記述や実行が容易なプログラミング言語を指す。また、「Java」はソースファイルをコンパイルし、実行ファイルを実行することでプログラムを走らせることが可能である。一方で「JavaScript」のプログラムは単体で動作させることができない。このように JavaScript と Java は名前は似ているが、この2つの言語は性質が異なるプログラミング言語である。

5 開発したシミュレータについて

5.1 教材に必要な機能

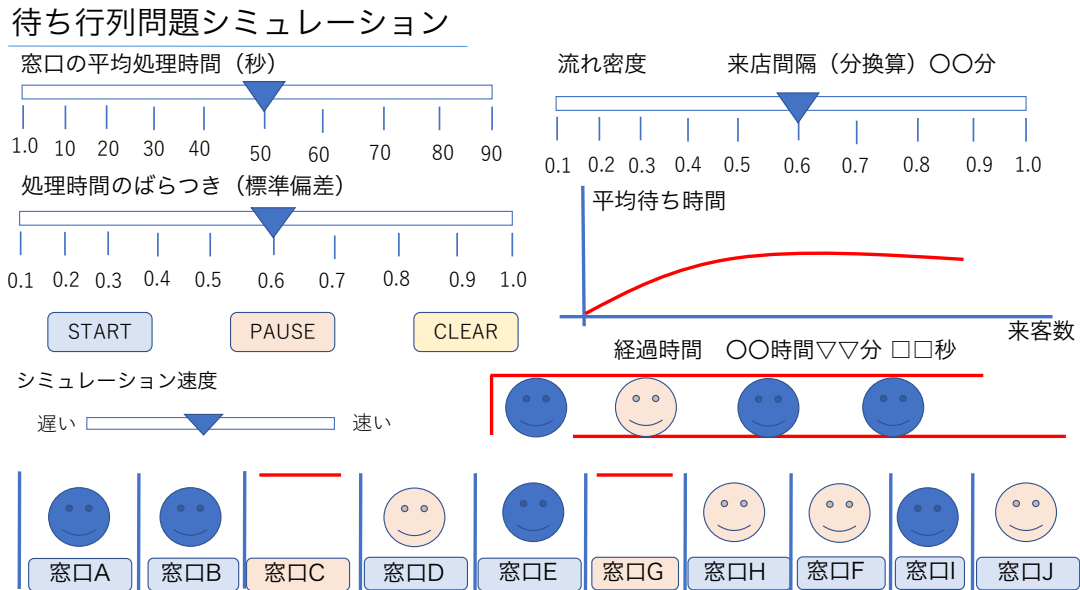


図9 待ち行列問題シミュレータ教材のレイアウト

この章では、待ち行列問題シミュレータ教材を開発する上で、学習者の理解度向上を考えどのような機能があるか説明していく。図9は開発する以前に作成した待ち行列問題シミュレータ教材のレイアウトである。

<待ち行列状態の表示>

本研究はアニメーションをより現実的な動きに寄せブラウザ上への移植及び改良することでより利用者の理解度の向上を目的としている。そのため今回開発する教材は過去の研究で作成されたものを使用した。基本的なUIはそのままに、アニメーション部の改良が本研究の大きな目標である。そうすることで、より現実に近い動きになりさらに理解度の向上が見込めると考えた。

<待ち行列状態を設定するパラメータの設定>

待ち行列の状態は、様々なパラメータの影響を受け変化する。そのパラメータとして、窓口の数、客の流れ密度、サービスに必要な平均時間、処理時間のばらつき（標準偏差）がある。これらパラメータを変化させた時、待ち行列の状態にどのような変化が起きるか、学習者に視覚的に理解させる必要がある。そのため、学習者がパラメータを設定できるようにした。

＜アニメーションについて＞

過去の研究はアニメーションが擬似的なものであり動きが少なく現実の動きと相違があることなどが問題であると考えた．そこで本研究では，**CSS・JavaScript** を用いたアニメーションで待ち行列問題のシミュレーションを行うこと目標とした．

＜アニメーションの再生・一時停止＞

アニメーションを行うことで，現象を視覚的に体験でき学習者の理解を促すことが可能である．今回作成した教材ではアニメーションをスタートさせる **START** ボタン，アニメーションを一時停止させる **PAUSE** ボタンを作成した．また過去の研究で作成されたものと同様に，アニメーションを実行中でもパラメータの変更を可能とし，パラメータの変化による待ち行列へ影響を視覚的に学ぶことができる．

5.2 開発した教材の実行例

前述であげた機能について，実行例をあげていく．図 11 は開発した待ち行列問題教材の画面である．

待ち行列問題シミュレータ

The screenshot shows the '待ち行列問題シミュレータ' (Queue Problem Simulator) interface. It features a top section with four sliders for parameter adjustment, a middle section with 'START' and 'PAUSE' buttons, and a bottom section for the simulation area. Red circles with numbers 1, 2, and 3 are overlaid on the interface to highlight specific features.

- ①** Points to the parameter sliders: '窓口の平均処理時間 (分): 1', '処理時間のばらつき (標準偏差): 0.1', '流れ密度: 0.1', and '窓口の数: 1'.
- ②** Points to the 'START' and 'PAUSE' buttons.
- ③** Points to the simulation area, which contains a horizontal row of ten empty rectangular boxes representing service windows.

図 10 開発した待ち行列問題シミュレータ教材のレイアウト

①では各パラメータの設定をユーザーによって変更できる．窓口の平均処理時間 (iii) δ_0 は 1～10 (分)，標準偏差 (iv) σ は 0.1～1.0，流れ密度 (i) α は 0.1～5.0，窓口の数は 1～10 の間で変更できる．また各パラメータはアニメーションを実行中にも変更が可能である．そのことにより各パラメータを変化させた時の状況遷移を視覚的に学びやすくなると考えた．





②ではアニメーションを開始・停止できるボタンを設置した．各パラメータを定め，**START** ボタンを押すことでアニメーションがスタートする．**PAUSE** ボタンは押すことでアニメーションの途中で止めることができる．

③はユーザーが設定したパラメータに応じて，アニメーションが実行される部分である．アニメーションにより現象をより直感的に理解できると考えた．

5.2.1 各パラメータについて

待ち行列の状態は，窓口数，窓口の平均処理時間，客の流れ密度，処理時間のばらつき（標準偏差）のパラメータが影響している．これらのパラメータを，図 11 ①部分のスライダーで設定する．

待ち行列問題シミュレータ

窓口の平均処理時間（分）： 1	流れ密度： 0.1
	
処理時間のばらつき（標準偏差）： 0.1	窓口の数： 1
	

START	PAUSE
-------	-------



図 11 設定するパラメータ

5.2.2 実行方法について

実行方法について図 12 に示す。前述した図 11 の各パラメータを設定し、**START** ボタンを押すことでアニメーションが開始する。図 13 は、窓口数:10、窓口の平均処理時間:6、客の流れ密度:0.8、処理時間のばらつき:0.1 に設定した画面である。図 13 は、実際にパラメータを設定し実行した際の画面を示している。

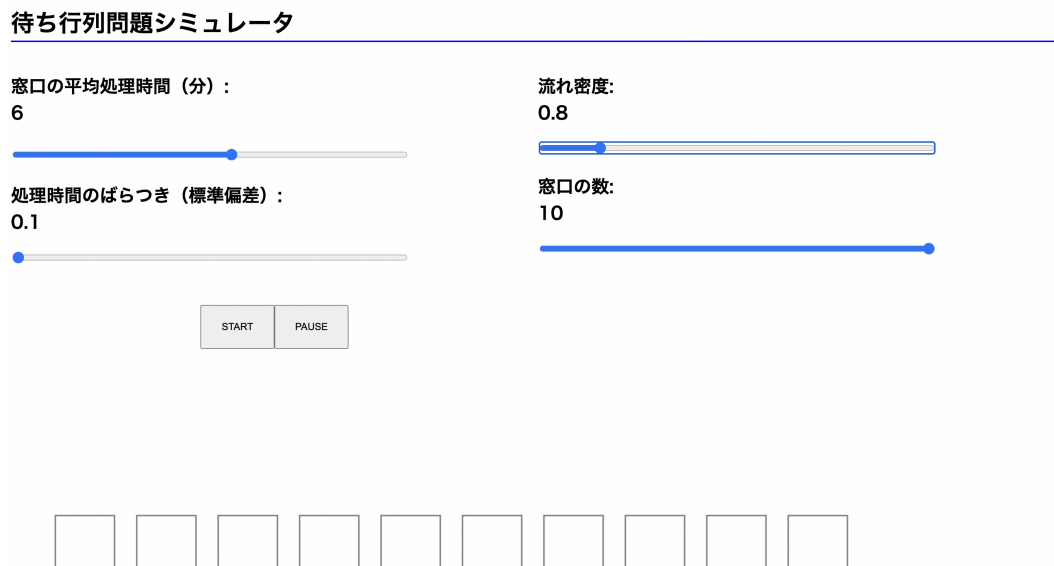


図 12 パラメータ設定後の画面

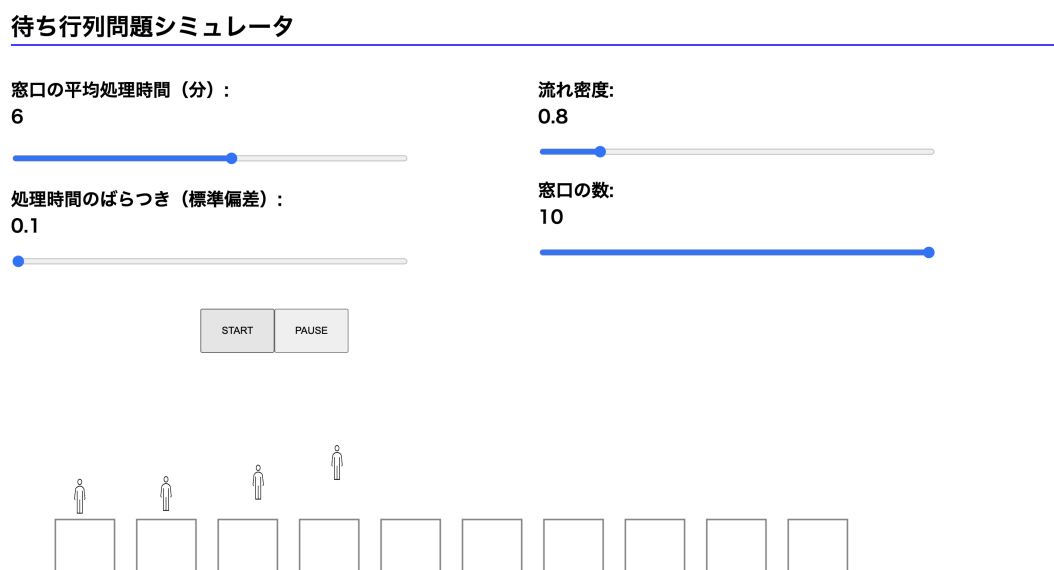


図 13 アニメーション実行例

5.3 開発したプログラムについて

ファイル一覧

sample.html シミュレータの表示部分，各パラメータのスライダーやアニメーションの行う位置を記述している．

sample.css HTML 内要素の配置，アニメーションの速度，移動の仕方などの設定を行う．

Customer.js 客を取り扱うクラスである．客が来たタイミングで客の画像を作成している．コンストラクタとして

- (1) δ - 客の処理時間の平均値
 - (2) σ - 処理時間の標準偏差
 - (3) *number* - 待ち行列内の順序
 - (4) *y*- 待ち行列の *y* 座標
- を定義した．

Figure.js 窓口を表す図形を作成する．

List.js 待ち行列を取り扱う．客が窓口に入った後の行列を詰める動きを記述している．コンストラクタとして

- (i) δ - 客の処理時間の平均値
 - (ii) σ - 処理時間の標準偏差
 - (iii) α - 客の流れ密度
- を定義した．

nylon.js アニメーションの開始・停止を記述している．

Register.js 各窓口を扱う．どの窓口を表しているか示す *id*，窓口前の座標をコンストラクタとして受け取り客を対応する窓口前へ移動させる．コンストラクタとして

- (I) *id*- どの窓口かを区別する *id*
 - (II) *x*- 窓口前の *x* 座標
 - (III) *y* - 窓口前の *y* 座標
- を定義した．

Shop.js 全体（客，窓口）を制御する．ユーザーが設定したパラメータを受け取り，窓口に入るタイミングを指定している．コンストラクタとして

- (a) *number*- どの窓口かを区別する *id*
- (b) δ - 客の処理時間の平均値
- (c) σ - 処理時間の標準偏差
- (d) α - 客の流れ密度

vbtimer.js アニメーションを連続的に動かしている．

waitinglist_ui.js 図 11 ②部分のボタンが押された時の処理を記述している．

クラス一覧

class Customer 客を扱うクラス.

class List 待ち行列を扱うクラス.

class nylon 受け取ったイベント情報をそのまま外部に渡すクラス.

class Register レジ（単独）を扱うクラス.

class shop 全体制御のためのクラス.

class vbTimer 内部的に時間を進める時の開始・停止の処理を扱うクラス.

メソッド一覧

move メソッド 客が窓口でサービスを受ける座標まで移動させている.

```
const regi_enter = { // 客が窓口でサービスを受ける座標
  x: [60, 170, 280, 390, 500, 610, 720, 830, 940, 1050],
  y: [650, 650, 650, 650, 650, 650, 650, 650, 650, 650]
}
this._img.style.left = `${regi_enter.x[id]}px`;
this._img.style.top = `${regi_enter.y[id]}px`;
```

process メソッド 内部的に時間を進める.

```
this._remain -= time;
return this._remain;
```

gauss メソッド 客の処理時間を設定する.

```
let del = 0.0;
for( let p=0; p<12; p++ ) {
  del += Math.random();
}
return delta + sigma * (del - 6.0);
```

getLength メソッド レジ数を返す.

```
return this._list.length;
```

shift メソッド 待ち行列の先頭を取り除く.

```
if (this._list.length == 0) throw new RangeError('待ち行列に客がいませ
return this._list.shift();
```

move_gyouretu メソッド 行列を移動させる. 先頭の客が窓口に入った時の列を詰める動きを記述している.

```
if (this._list.length == 0) throw new RangeError('待ち行列に客がいませ
return this._list.shift();
```


poison メソッド 次の客が来るまでの時間を設定する.

```
return -Math.log( Math.random() ) / this._alpha;
```

setPassThrough メソッド 受け取ったイベント情報をそのまま外部に渡すための関数である.
nylon.iFrame で使用する.

```
if( window.nylon.passthrough == null ) {  
    window.nylon.passthrough = fn;  
} else {  
    console.log( "Error by duplexy registration" );  
}
```

on メソッド キーワードと、キーワードに対する処理を登録する関数である.

```
console.log( this );  
if( this.map[ keyword ] == null ) {  
    // if( window != parent ) {  
    //     this.parent.postMessage( { "keyword": ["on"],  
    // }  
    console.log( "new keyword : " + keyword );  
    this.map[ keyword ] = [ new nylonfunc( fn, this ) ];  
} else {  
    this.map[ keyword ].push( new nylonfunc( fn, this ) );  
}
```

emitByArray メソッド イベントを起こす関数 1 である.

```
if( window.nylon.passthrough != null ) {  
    window.nylon.passthrough( keys, params, this );  
}  
if( params == null ) {  
    params = {};  
}  
console.log( "→" + params["keywords"] );  
  
if( params["keywords"] == null ) {  
    console.log( "params.keywordsがnull");  
    params["keywords"] = keys;  
}
```

```

    for( let key of keys ) {
        if( this.map[ key ] == null ) {
            console.log( "Invarid keyword " + key );
        } else {
            for( let element of this.map[ key ] ) {
                //console.log( element );
                //console.log( "—>" + element.obj );
                if( element.obj != this ) {
                    element.fn( key, params );
                }
            }
        }
    }
}

```

emit メソッド イベントを起こす関数 2 である.

```

var keys = keyword.split( "|" );
this.emitByArray( keys, params );

```

isProgres メソッド 客に対応中か確認する. 対応中: true, 空き状態: false を返す.

```

return this._progress != null;

```

push メソッド レジに客を誘導して処理を開始する.

```

if ( this.remain < 0 ) customer.process(-this.remain);
else customer.process(-rew);
this._progress = customer;
customer.img.style.left = this.x;
customer.img.style.top = this.y;

```

enable メソッド 内部的に時間を進める時の開始を定義している.

```

this._tHandle = setInterval( this.timer, this.interval );
this.enabled = true;

```

disable メソッド 内部的に時間を進める時の停止を定義している.

```
if( this.enabled == true ) {  
    clearInterval ( this._tHandle );  
    this.enabled = false;  
}
```

nl2.on メソッド START ボタンが押された時に enable メソッド, PAUSE が押された時 disable メソッドを呼び出す.

```
timer.enable();  
timer.disable();
```

6 おわりに

本研究では、先行研究をもとに視覚的・直感的に理解できることを目的とした待ち行列問題シミュレータをブラウザへ改良を行い、移植することを目的とした。

問題点として、先行研究は開発環境が古く現在は使用できないこと、アニメーションが画像を切り替えていくことによる擬似的なアニメーションであり、なめらかなアニメーションではなかったという点がある。

結果として、先行研究と比較してアニメーションの部分を改良することができた。具体的には画像を切り替えていくことによる擬似的なアニメーションではなく、人の画像を座標を指定して実際に窓口の前まで動かし、窓口から退場の動きも付け加えることができた。

本教材が科目「数値計算」における「待ち行列問題」を理解する上で少しでも役に立つことを期待している。

7 謝辞

本研究の遂行及び本論文作成にあたり，多大なるご指導，ご助言を頂きました 須田 宇宙准教授に深く感謝の意を表します．

参考文献

- [1] 平成 16 年度 卒業論文 モンテカルロ法による待ち行列問題のマルチメディア教材 (薄井英彦・梅山卓也)
- [2] 総務省統計局 最適な設備投資を把握” , https://www.stat.go.jp/naruhodo/15_episode/toukeigaku/setubitoshi.html, 2022/12/5 参照
- [3] HTML クイックリファレンス” , <http://www.htmq.com/>, 2022/12/5 参照
- [4] HTML・CSS のそれぞれの基礎知識と違いとは? ” , <https://www.sedesign.co.jp/blog/html-css>, 2022/12/6 参照
- [5] Web 制作に欠かせない CSS とは? 役割や活用メリット、使い方などを解説します! ” , https://persol-tech-s.co.jp/hatalabo/it_engineer/502.html, 2022/12/6 参照
- [6] JavaScript とは? modis ” , https://www.modis.co.jp/candidate/insight/column_29, 2022/12/6 参照
- [7] 指数分布の意味と具体例” , <https://manabitimes.jp/math/1006>, 2022/12/30 参照