0
Рандомізовані алгоритми

Мета: 1) уяснити роль випадковості у фізиці;
2) отримати уявлення про рандомізовані алгоритми та закони розподілу випадкових величин.

0.1 Випадковість у фізиці

Випадкова величина (англ. random variable) — величина, можливими значеннями якої є результати випробувань чи спостережень явищ або процесів, що носять випадковий характер. Класичними прикладами випадкової величини є результати підкидання монети, кубика із числовими позначеннями на його гранях, швидкості руху частинки при броунівському русі тощо. Випадкова величина – одне з основних понять теорії ймовірностей. Випадковою величиною можна назвати будь-яку (не обов'язково чисельну) змінну x, значення

якої утворюють множину випадкових елементарних подій $\{x\}$. Якщо випадкова величина подається у вигляді функції, то вона повинна бути вимірною (тобто вона не може мати точок розривів та невизначеностей, ніде не може приймати значення $\pm\infty$).

Зазвичай результати випробувань, які є випадковими величинами, залежать від деяких фізичних змінних, які не мають чіткої визначеності. Наприклад, при підкиданні звичайної монети, кінцевий результат (впаде вона аверсом чи реверсом) залежить від невизначених фізичних параметрів. Який результат буде зрештою спостерігатися є непевним. Областю визначення випадкової величини є множина можливих результатів. У випадку з монетою, розглядають лише два можливих результатом – вона впаде однією з двох сторін.

Випадкова величина визначається як функція, яка відображає результати у вигляді числових величин (міток), що зазвичай задаються дійсними числами. Повинна існувати процедура присвоєння чисельного значення кожному можливому результату експерименту, і, на відміну від іменування назвами, ця процедура сама по собі не є випадковою і не є змінною. Те що є випадковим – це неусталений фізичний процес,

який описує як падає монета, і невизначеність результату, який буде спостерігатися в конкретний момент.

У фізиці випадкові величини – це або результати вимірювання чогось, або флуктуації. Результати вимірювання фізичної величини є випадковими, оскільки як засоби вимірювання, так і методи вимірювання фізичних величин мають певну обмежену точність. Флуктуація – це випадкове відхилення значення фізичної величини від середнього в певній ділянці простору чи в певний момент часу. Флуктуації відіграють велику роль у різних фізичних процесах. Характерним прикладом фізичного явища, в якому визначальними є флуктуації його характеристик є броунівський рух. Він зумовлений флуктуаціями тиску, що діють на частинки розмірів порядку броунівських.

0.2 Програмне середовище для практичних робіт

Для даних практичних робіт буде використовуватися мова програмування JavaScript. Такий вибір обумовлений насамперед її доступністю та

справжньою крос-платформеністю, оскільки вона виконується прямо в браузері. Для написання коду можна використовувати будь-який текстовий редактор, але перевагу слід надати тому, який вміє підсвічувати синтаксис JavaScript (Notepad++, Sublime Text тощо).

В ОС Windows подібну розробку зручно вести за допомогою Open Server, але ми будемо працювати лише з фронтовою частиною, бекову ми чіпати не будемо. Проте Open Server дає зручні можливості для швидкого запуску проектів.

Спочатку його потрібно встановити. Open Server є вільним програмним забезпеченням. Його можливо завантажити безкоштовно або з офіційного репозиторію https://ospanel.io/download/ або зі сторонніх сайтів на кшталт https://biblprog.org.ua/ua/open_server/. Його рекомендується ставити у кореневий каталог C:\ . Після встановлення його слід запустити, і на панелі фонових задач (праворуч) повинен з’явитися червоний прапорець. Натиснувши по ньому правою кнопкою миші, відкривається меню Open Server – рис. 0.1.

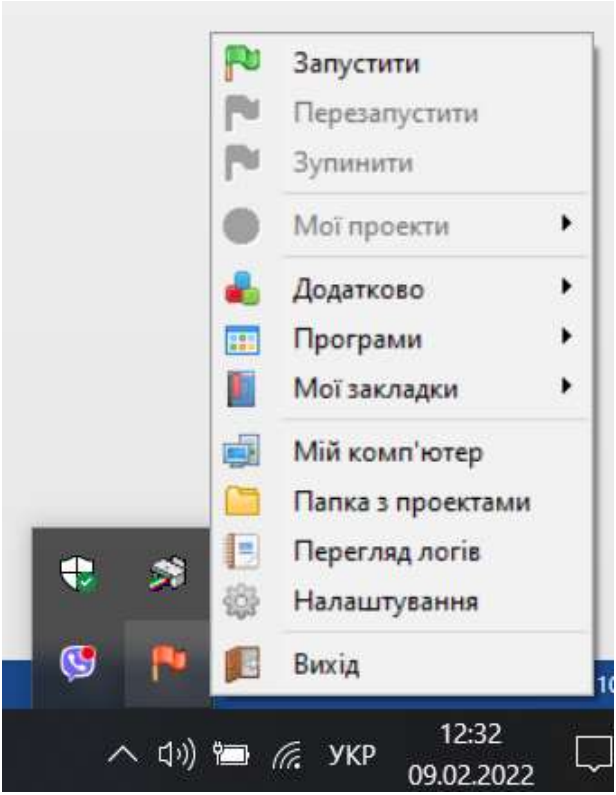

Рис. 0.1 – Меню Open Server

Перевіряємо, чи все встановилося правильно. Для цього запускаємо сервер (зелений прапорець – «Запустити»). Прапорець на панелі завдань також повинен стати зеленим. Тоді заходимо в «Мої проекти» і вибираємо «localhost» – рис. 0.2. Повинно відкритися вікно браузера з повідомленням про те, що Open Server працює. У версії Open Server 5.4.1 (актуально станом на лютий 2022 року)) це буде Opera (але, наприклад, у

версії 5.2 і раніше це був Chrome… Там розробники якось міняють всю збірку – мабуть, це якось пов’язано з якоюсь інтелектуальною власністю чи щось таке…

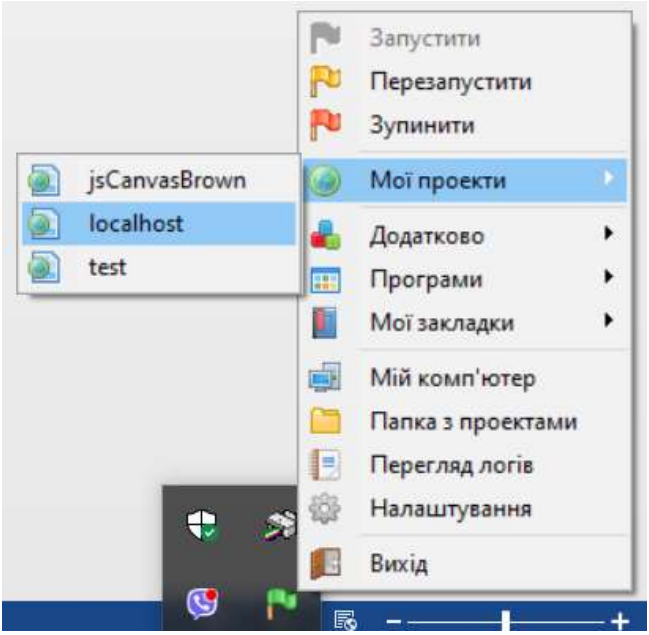

Рис. 0.2 – Запуск якогось проекту в Open Server

Якщо такого немає – значить, ви неправильно встановили Open Server, або встановили його поверх попередньої версії (а так не можна, він не оновлюється, а перевстановлюється заново), або щось напортачили у налаштуваннях (не треба було туди взагалі лізти). В такому випадку найпростіше всього видалити Open

Server і перевстановити заново. Заодно не завадить почитати RTFM по ньому: https://ospanel.io/docs/

В будь-якому випадку ви зможете подальші проекти запускати у своєму улюбленому браузері.

Тепер створюємо новий проект. Для цього треба зайти у «Налаштування» – відкриється нове вікно, в якому слід вибрати вкладку «Домени» – рис. 0.3.

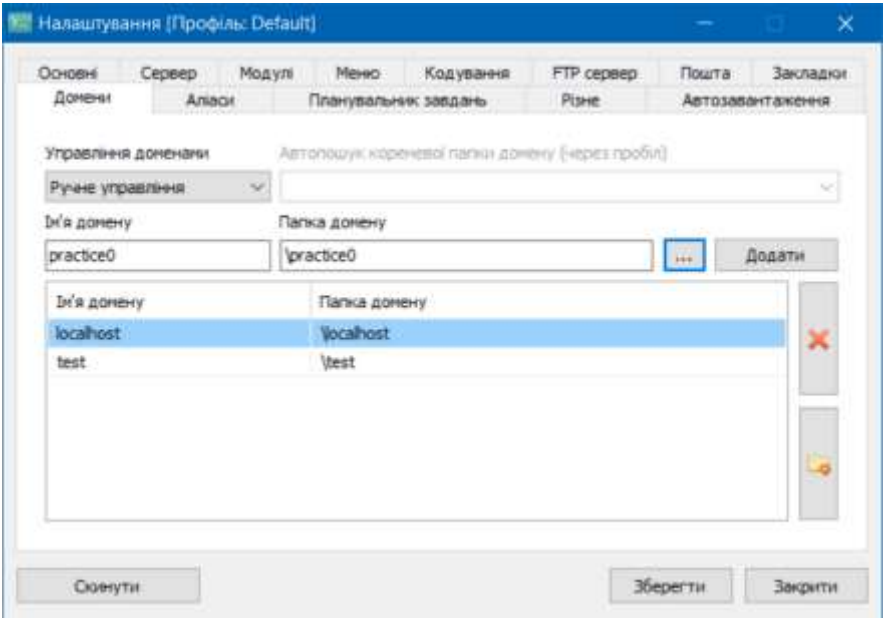

Рис. 0.3 – Вікно налаштувань Open Server

Там, де «Управління доменами» – слід виставити «Ручне управління». В рядку нижче слід вписати ім’я домену, а потім натиснути кнопку «…» і створити папку

домену. Зверніть увагу, що в імені домену не допускаються символи « » (пробіл) та «_», а також вводити в ім'я домену всякі символи на кшталт +, -, *, /, =, #, лапки, $, ?, !, :, <, >, всякі дужки тощо і кирилицю – це дуже погана ідея. Краще, щоб папка домену називалася так само, це позбавить від зайвої плутанини потім. Після всіх цих маніпуляцій слід натиснути кнопку «Додати» – новий домен повинен з'явитися у списку нижче. В кінці нажимаємо «Зберегти», сервер каже, що йому треба перезавантажитись – даєте йому згоду на це дійство (перезавантажиться лише Open Server, а не комп'ютер), закриваєте це вікно.

Тепер можна запускати редактор коду. Якщо у вас є свій улюблений, запускайте його, якщо ні, в Open Server вбудований Sublime Text (Open Server → Програми → Sublime_text).

Оскільки для візуалізації кінцевого результату ми будемо використовувати браузер, то слід нагадати про загальну структуру веб-сторінки. В найпростішому випадку вона складається з файлу `index.html`, а також файлу стилів `styles.css` та файлів `*.js` з JavaScript-кодом. Ваші поточні веб-сторінки у даному випадку зберігаються у папці `C:\OpenServer\domains`. Наприклад, папка цієї практичної роботи буде

називатися practice0. Тоді вам потрібно спочатку створити в редакторі коду пустий документ, набрати в ньому найпростішу заготовку пустої веб-сторінки та зберегти його під іменем index.html у папці C:\OpenServer\domains\practice0\

Заготовка:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet"
          href="styles.css"
          type="text/css">
    <title>Practice 0</title>
  </head>
  <body>
    <header>
      <script type="text/javascript"
              src="whowroteit.js">
      </script>
    </header>
    <div class="container">
    </div>
  </body>
</html
```

Тепер опишемо стилі (styles.css):

```
header {
```

```
  border: 1px solid black;
}

.container {
  display: grid;
  border: 1px solid black;
}
```

Щодо файлу whowroteit.js, то це буде заголовок-преамбула, який буде вказувати на те, хто писав цей код (це типу повинно бути захистом від намагання здати чужий код від свого імені…). Правила «хорошого тону» в сучасній веб-розробці говорять, що не слід змішувати HTML, CSS, JavaScript і мову, на якій пишуть бек (PHP, RoR, nodeJS чи щось подібне) в одному файлі – тому ми теж будемо дотримуватися цього правила. Отже, whowroteit.js:

```
whoWroteIt(%lessonNumber%, %lessonTheme%,
           %studentName%, %studetnGroup%)

//=============================================

function whoWroteIt(lessonNumber, lessonTheme,
                    studentName, studetnGroup) {
  document.write("Практичне заняття " +
                 lessonNumber);
```

```
  document.write("<h2>" + lessonTheme + "</h2>");
  document.write("<p>" + studentName + "</p>");
  document.write("<p>" + studetnGroup + "</p>");
}
```

Зверніть увагу, що замість `%lessonNumber%`, `%lessonTheme%`, `%studentName%`, `%studetnGroup%` ви підставляєте номер практичної роботи та її назву, своє прізвище та ім'я, і шифр академічної групи. Також зверніть увагу на те, що викликати функцію можна як до її опису, так і після. Ми будемо в кінці файлу описувати функції, а на початку файлу писати основний код.

Ця заготовка залишається на всі практичні роботи.

До кожної конкретної практичної роботи весь подальший код пишеться у js-файлі, який за допомогою тегу `<script>` підключається у контейнер. Наприклад, для цієї практичної роботи:

```
<div class="container">
  <script type="text/javascript" src="randoms.js">
  </script>
</div>
```

0.3 Генерація випадкових чисел

Найпростіший спосіб отримати випадкове число – це метод `Math.random()`, вбудований у JavaScript.

`Math.random()` завжди повертає число з плаваючою комою між 0 та 1. Технічно число, яке ви отримаєте, може бути 0, але ніколи не буде точно 1. Це все добре, але на практиці частіше потрібно отримати випадкове число у деякому діапазоні. Окрім того, це число часто повинно бути цілим. Для цього у файлі `randoms.js` пропишемо таку функцію:

```
function getIntRandomInRange(min, max) {
  return Math.floor(Math.random()*(max-min+1)+min);
}
```

Для перевірки того, що все працює коректно, поки що `randoms.js` буде таким:

```
document.write("<p>" + getRandom(0, 10) + "</p>");
document.write("<p>" + getIntRandomInRange(0, 10) +
"</p>");

//=====================================================

function getRandom(min, max) {
  return Math.random()*(max-min)+min;
}
```

```
function getIntRandomInRange(min, max) {
  return Math.floor(Math.random()*(max-min+1)+min);
}
```

Тепер можна запустити проект через Open Server Panel (в Opera) або у своєму браузері, запустивши `index.html`. Якщо все було зроблено правильно, повинен відобразитися header та 2 випадкових числа – дробове та ціле. Пообновлюйте кілька разів сторінку, пересвідчіться, що все працює, числа міняються.

0.4 Генерація випадкового масиву

Тепер перші два рядки у файлі `randoms.js` можна закоментарити (і таким чином прибрати їх з виконуваного коду), і під ними набрати таке:

```
let numPool = [];
let i;
for (i=0; i<101; i++) {
  numPool[i] = getIntRandomInRange(0, 100);
  document.write("<p>" + numPool[i] + "</p>");
}
```

Таким чином, буде створений масив `numPool` із 100 випадкових чисел. Пообновлюйте сторінку,

пересвідчіться в тому, що кожен раз масив забивається новими випадковими числами.

Але сам по собі масив випадкових чисел – річ малоцікава. Щоб щось можна було зробити з випадковими числами, потрібно знати їх закон розподілу. Вважається, що `Math.random()` генерує випадкові числа з рівномірним розподілом. Перевіримо це.

Для такої перевірки нам потрібно мати багато однорідних масивів. Взагалі вся теорія ймовірності працює лише на *великій кількості* подій, якщо ж кількість подій невелика, то оцінювати ймовірність у її класичному розумінні не можна. Тому будемо міркувати таким чином:

1) У масиву є індекс. Нехай цей індекс і буде випадковим числом.
2) Заповнюємо масив нулями – тобто поки що жодного випадкового числа не згенерували.
3) Генеруємо випадкове ціле число. Збільшуємо на 1 відповідний елемент масиву.
4) Повторюємо цю операцію достатньо багато разів. «Достатньо багато» означає, що якщо, наприклад, масив має 101 елемент (числа від 0 до 100), то ми повинні згенерувати не менше 1000 чисел – щоб

кожне із чисел від 0 до 100 згенерувалось принаймні кілька разів. Тоді значення кожного елементу масиву буде показувати, *скільки разів* згенерувалося відповідне число.

Тепер спробуємо запрограмувати цей алгоритм. Куди конкретно це писати – ви повинні визначити самі.

```
let numPool = [];
let i;

let iterations = 10000;
let tempNumber;
let min = 0;
let max = 100;

for (i=0; i<max+1; i++) {
  numPool[i] = 0;
}

for (i=0; i<iterations; i++) {
  tempNumber = getIntRandomInRange(min, max);
  numPool[tempNumber]++;
}

for (i=0; i<max+1; i++) {
  document.write("<p>"+i+"    "+numPool[i]+"</p>");
}
```

Тепер запустіть це і пересвідчіться, що всі елементи масиву *майже* однакові – тобто якщо ви згенерували 10000 випадкових чисел, то *в середньому* кожен елемент масиву дорівнює

$$\frac{\text{кількість згенерованих випадкових чисел}}{\text{кількість елементів у масиві}}$$

Таким чином робимо висновок про те, що Math.random() генерує числа з практично рівномірним законом розподілу.

0.5 Візуалізація закону розподілу випадкових чисел

Кожен раз дивитися на потоки чисел – це не цікаво. Одна картина важить тисячі слів – тому спробуємо візуалізувати наш випадковий масив. Для побудови зображень «на льоту» в HTML5 є елемент `canvas`. Підключимо його (у файлі `index.html`):

```
<div class="container">
  <canvas id="tutorialPDF" width="103" height="300">
  <!-- Probability Distribution Function !-->
  <script type="text/javascript" src="randoms.js">
```

```
  </script>
</canvas>
```

Наш `<canvas>` зараз називається `tutorialPDF` – «tutorial» тому, що ми тільки вчимося з ним працювати, а «PDF» – тому що ми на нього будемо виводити те, що в теорії ймовірностей називається «функція розподілу ймовірності» – на англійській мові цей термін звучить як «**P**robability **D**istribution **F**unction» (скорочено PDF). Крім того, ще є термін «функція щільності ймовірності», який в англомовній літературі звучить як «**p**robability **d**ensity **f**unction» (скорочено pdf). Між ними є суттєва різниця, але її пояснення виходить за рамки даної теми, тому поки що ми просто зупинимося на PDF.

Ширина `canvas` взята 103 пікселя, бо наш масив має 101 елемент, і ще 2 пікселя взяті «про запас». Висота 300 пікселів – теж «із запасом».

Для візуалізації пропишемо у `styles.css` таке:

```
#tutorialPDF {
    display: inline-grid;
    border: 1px solid black;
}
```

а у `randoms.js` таке:

```
let canvas = document.getElementById('tutorialPDF');
```

```
if (canvas.getContext) {
  let ctx = canvas.getContext('2d');
  let x0 = 0;
  let y0 = 200;
  for (i=0; i<max+1; i++) {
    ctx.beginPath();
    ctx.moveTo(x0+i, y0);
    ctx.lineTo(x0+i, y0-numPool[i]);
    ctx.closePath();
    ctx.stroke();
  }
}
```

Запустивши код, отримаємо зображення нашого закону розподілу – щось схоже на рис. 0.4.

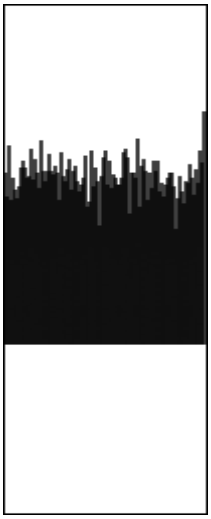

Рис. 0.4 – Приклад [квазі]-рівномірного розподілу

Можна кілька разів пообновлювати сторінку – побачити, що кожен раз картинка розподілу змінюється, але при цьому її загальний характер залишається незмінним.

0.6 Нормальний закон розподілу випадкової величини

Це все прекрасно, але у природі рівномірний закон розподілу випадкової величини трапляється вельми рідко. Природа найбільше любить *нормальний закон розподілу випадкової величини*. Строго кажучи, такий закон розподілу має два параметри – математичне сподівання (або середнє значення) та дисперсію (або середньоквадратичне відхилення). Нормальний закон розподілу (або його ще називають законом розподілу Гауса) виражається формулою:

$$f(x;\ \mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

На рис. 0.5 показані приклади нормального закону розподілу при різних значеннях μ і σ.

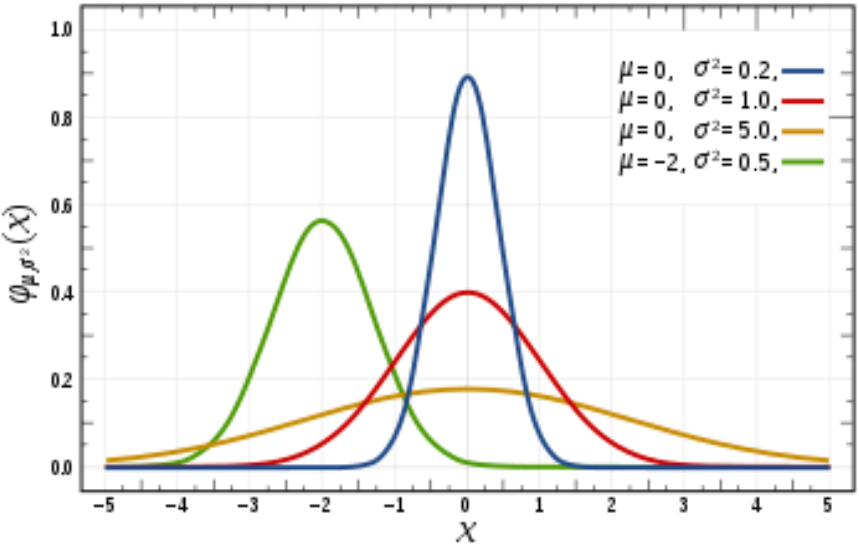

Рис. 0.5 – Приклади нормальних розподілів

Реалізація засобами JavaScript справжнього нормального розподілу – вельми нетривіальна задача. Але, на щастя, в математиці давно доведено, що нормальний закон розподілу – це гранична сума рівномірних законів розподілу. Тому генерація випадкової величини, що має нормальний закон розподілу у найпростішому випадку може бути зроблена як сума кількох випадкових величини (кожна з яких має *рівномірний* закон розподілу), поділена на їх кількість (таке собі усереднення):

$$X_{norm} = \frac{X_{1_{uniform}} + X_{2_{uniform}} + \cdots + X_{N_{uniform}}}{N},$$

причому чим більшим буде N – тим більше кінцевий результат буде схожий на нормальний закон розподілу.

Спробуємо по цьому принципу реалізувати нормальний закон. Для цього у файлі `randoms.js` закоментаримо рядок

```
tempNumber = getIntRandomInRange(min, max);
```

і замінимо його на таке:

```
tempNumber = (getIntRandomInRange(min, max) +
              getIntRandomInRange(min, max) +
              getIntRandomInRange(min, max))/3;
```

Зверніть увагу, що в кінці потрібно поділити на стільки, скільки разів ви викликаєте функцію `getIntRandomInRange(min, max)`.

Тепер візуалізуйте отриманий закон розподілу, і подивіться як змінюється його загальний характер в залежності від N (рис. 0.6)

Строго кажучи, те що ми отримали не є абсолютно точним нормальним законом розподілу, оскільки ми ніде в явному вигляді не задавали ні μ, ні σ. Проте воно вже точно не є рівномірним законом розподілу і достатньо схоже на нормальний.

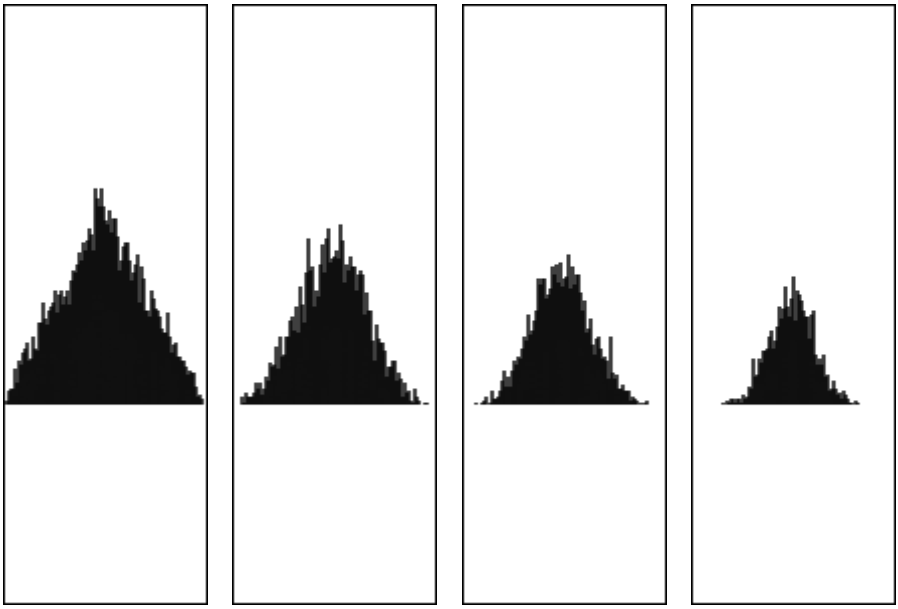

Рис. 0.6 – [Квазі]-нормальні розподіли ($N = 2, 3, 4, 7$)

0.7 Зміст звіту

Звіт з цієї практичної роботи повинен включати наступне:

0) Титульна сторінка.
1) Назва і мета практичної роботи.
2) Лістинг остаточного варіанту файлу `index.html` із заповненими тегами `<header>`, `<div>` та `<canvas>`, а також тегами `<script>` із включеннями файлів `whowroteit.js` та `randoms.js`.
3) Лістинг остаточного варіанту файлу `styles.css`.

4) Лістинг файлу `whowroteit.js`.
5) Лістинг проміжного варіанту файлу `randoms.js`, який виводить на екран випадкові числа (масив `numPool[]`) та скріншот вікна браузера із початком цього масиву.
6) Лістинг остаточного варіанту файлу randoms.js, який виводить квазі-рівномірний та квазі-нормальний розподіли випадкових чисел (якийсь один рядок у коді – що відповідає або за рівномірний, або за нормальний розподіл – може бути закоментарений).
7) Скріншот вікна браузера із прикладом графіка квазі-рівномірного закону розподілу.
8) Скріншот вікна браузера із прикладом графіка квазі-нормального закону розподілу.
9) Висновки. Чому перший закон розподілу називається «рівномірним»? Як зробити ваш квазі-рівномірний закон розподілу більш рівномірним? Чому дорівнює середнє у вашого нормального закону розподілу? Що може бути, якщо перед початком формування закону розподілу не обнулити примусово масив?

Звіт з практики зберегти у форматі pdf і надіслати на пошту krt_kro@ztu.edu.ua.

0.8 Контрольні запинання

1) Що таке випадкова величина?
2) Які ви можете навести приклади випадкових величин у реальних фізичних процесах?
3) Що таке закон розподілу випадкової величини?
4) Що таке рівномірний закон розподілу випадкової величини?
5) Чим характеризується рівномірний закон розподілу випадкової величини?
6) Чому JS генерує квазі-рівномірний розподіл?
7) Як зробити квазі-рівномірний закон розподілу випадкової величини більш рівномірним?
8) Що таке нормальний закон розподілу випадкової величини?
9) Які ви знаєте числові характеристики нормального закону розподілу?

Ускладнені питання:

1) Чому на рис. 0.6 для випадку $N = 7$ початок і кінець розподілу пусті?
2) Поясніть алгоритм генерації випадкового числа у діапазоні $[min, max]$ при тому, що штатними засобами мови програмування генерується випадкове число у діапазоні $[0, 1)$.