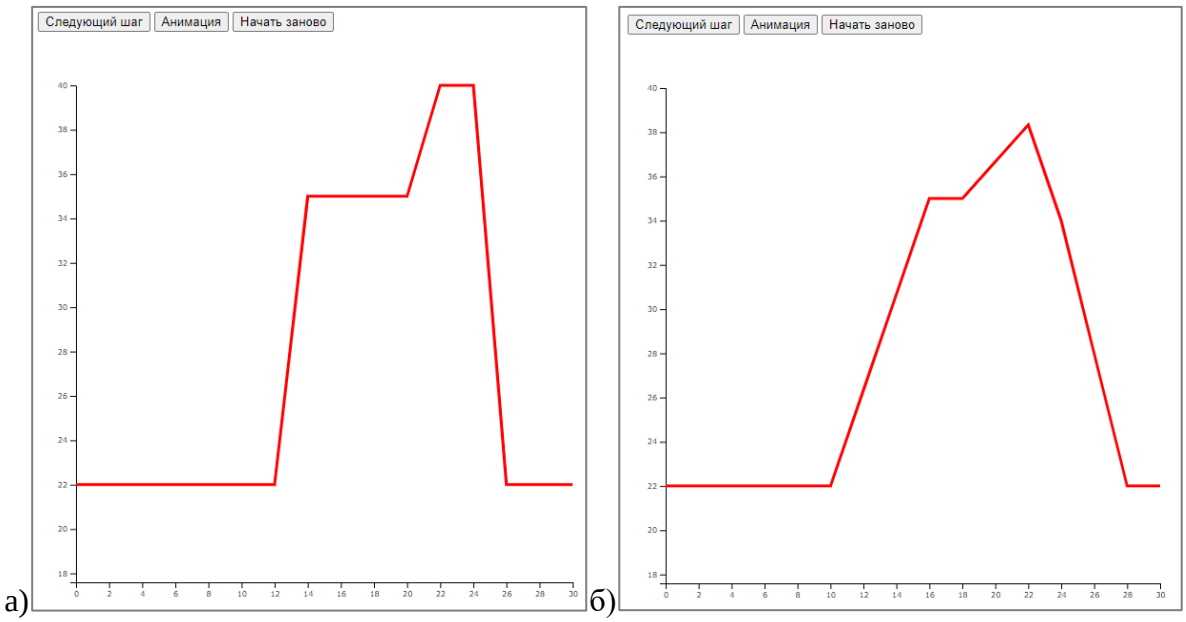
Лабораторная работа
«Построение графиков и их анимация, библиотека D3»

Задание. Создать html-страницу, которая включает (рисунок 1):

- форму для управления анимацией;
- область **`svg`** для вывода и анимации графиков.



а) б)

Рисунок 1. Страница HTML с графиком при t=0 и t=1

График формируется на основе данных из файла **`dataTemp.js`**. Фрагмент данных:

```
dataTemp = [
    {t:0, data:
        [{x:0, y:22.0},
        {x:2, y:22.0},
       ...
        {x:28, y:22.0},
        {x:30, y:22.0},
    ]},
    {t:1, data:
        [{x:0, y:22.0},
       ...
        {x:30, y:22.0},
    ]},
       ...
]
```

Пояснение. В массив **`dataTemp`** занесено численное решение задачи о теплопроводности стержня по времени для **`t`** от 0 до **`n`**. Стержень длиной 30 см имеет температуру 22 градуса. В начальный момент времени (**`t=0`**) стержень нагревается до температуры 35 градусов на одном участке и до 40 градусов на другом (рисунок 1а). С течением времени стержень начинает остывать. Например, в момент времени **`t=1`** на рисунке 1б показано распределение температур по стержню. Температура внешней среды сохраняется и имеет температуру 22 градуса во время всего процесса остывания.

При клике по кнопку **Начать** и при загрузке страницы должен строиться динамический график по данным из массива при **t=0** (рисунок 2)

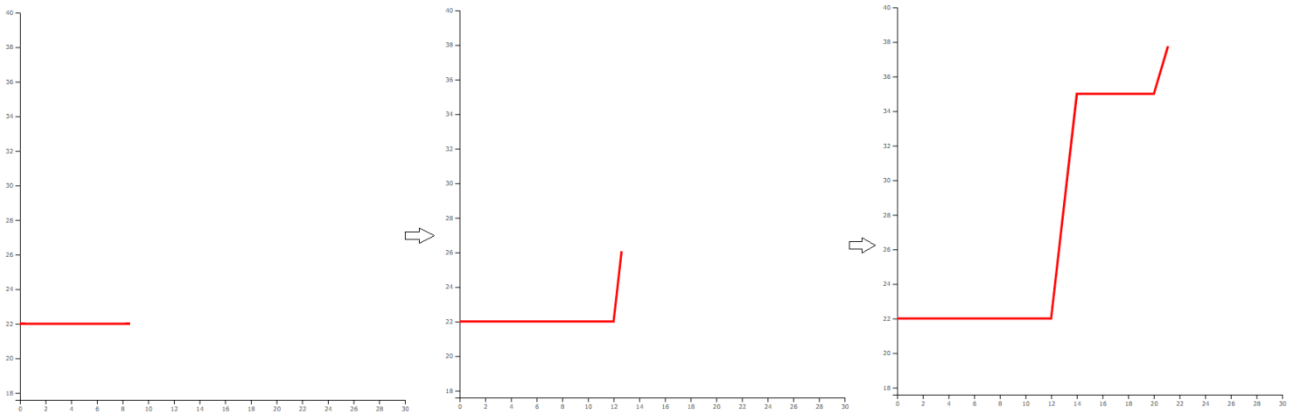


Рисунок 2. Динамическое построение графика

При клике по кнопке **Следующий шаг** должен показываться переход от текущего состояния (момент времени **t**) к следующему (момент времени **t+1**) с использованием анимации. Вид графиков при **t=0** и **t=1** показан на рисунке 1.

По клике по кнопке **Анимация** должна показываться динамика изменения температур на стержне от текущего момента времени до последнего (**t=n**).

Порядок выполнения работы

1. Создать HTML-файл следующей структуры:

```
<html >
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="style.css">
  <script src="http://d3js.org/d3.v7.min.js"> </script>
</head>
<body>

   <form>
       <input type="button" value="Следующий шаг" id="step"> </input>
       <input type="button" value="Анимация" id="all"> </input>
       <input type="button" value="Начать заново" id="start"> </input>
   </form>
  <svg></svg>
</body>
  <script src="dataTemp.js"></script>
  <script src="program.js"></script>
</html>
```

2. В файл **style.css** добавить описание стилей для текста и осей на графике.

```
svg text {
    font: 8px Verdana;
}
svg path, line {
    fill: none;
    stroke: #333333;
}
```

3. Создать файл **program.js** и включить в него настройку параметров и инициализацию **svg**-элемента:

```
const width = 600;
const height = 600;
let marginX = 40;
let marginY = 40;

let svg = d3.select("svg")
    .attr("width", width)
    .attr("height", height) ;
```

4. Реализовать функцию **createAxes()**, которая в качестве параметра получает массив с данными для графика. Функция создает шкалы для преобразования значений массива к значениям в области построения и рисует оси координат. Результат выполнения функции – шкалы преобразования по оси OX и OY.

```
function createAxes(data) {

    let [min, max] = d3.extent(data.map(d => d.y));

    // функция интерполяции значений на оси
     let scaleX = d3.scaleLinear()
                    .domain(d3.extent(data.map(d => d.x)))
                    .range([0, width - 2 * marginX]);

     let scaleY = d3.scaleLinear()
                    .domain([min * 0.8, max])
                    .range([height - 2 * marginY, 0]);

     // создание осей
    let axisX = d3.axisBottom(scaleX); // горизонтальная
    let axisY = d3.axisLeft(scaleY); // вертикальная

    // отрисовка осей в SVG-элементе
    svg.append("g")
       .attr("transform", `translate(${marginX}, ${height - marginY})`)
       .call(axisX);

    svg.append("g")
       .attr("transform", `translate(${marginX}, ${marginY})`)
       .call(axisY);
       return [scaleX, scaleY]
}
```

5. Реализовать функцию **createPath()**, которая в качестве параметра получает шкалы преобразования по осям координат. Функция создает линию для вывода точек массива на график и добавляет графический примитив **path** в svg-область для отображения графика. Результат выполнения функции – шаблон для создания линии.

```
function createPath(scaleX, scaleY) {

    let lineXY = d3.line()
        .x(d => scaleX(d.x))
        .y(d => scaleY(d.y));

    svg.append("path") // добавляем путь
       .attr("id", "graph")
       .attr("transform", `translate(${marginX}, ${marginY})`)
```

```
        .style("stroke-width", "3")
        .style("stroke", "red");

    return lineXY;
}
```

6. Реализовать функцию **`firstStep()`**, которая в качестве параметра получает данные для построения двумерного графика и шаблон для создания линии по точкам. Функция выводит динамический график (линия постепенно рисуется в области построения).

```
function firstStep(data, line) {

    const firstPath = svg
        .select("path#graph")
        .datum(data)
        .attr("d", line);

     const pathLength = firstPath.node().getTotalLength();

     firstPath
        .attr("stroke-dashoffset", pathLength)
        .attr("stroke-dasharray", pathLength)
        .transition()
        .ease(d3.easeLinear)
        .duration(5000)
        .attr("stroke-dashoffset", 0);
}
```

7. Вывести в svg-элемент график первого шага (при **`t=0`**) при загрузке страницы.

```
let [scaleX, scaleY] = createAxes(dataTemp[0].data);

let lineXY = createPath(scaleX, scaleY);

firstStep(dataTemp[0].data, lineXY);

// следующий наг 1
let currentIndex = 1;
```

В результате будет построен график, показанный на рисунке 2.

8. С событием click кнопки **Следующий шаг** связать обработчик, в котором выводится распределение температур для элемента массива при **`t=currentIndex`**:

```
d3.select("#step")
.on ('click', () => {

        if (currentIndex >= dataTemp.length -1) return;

        update(dataTemp[currentIndex].data, lineXY);

        currentIndex++;
    }
);
```

9. Реализовать функцию **`update()`**, которая в качестве параметра получает данные для построения, шаблон преобразования точек и время анимации. Функция осуществляет анимированный переход от одного шага к другому.

```
function update(data, line, time=1000) {

      svg.select("path#graph")
         .datum(data)
         .transition()
         .duration(time)
         .ease(d3.easeLinear)
        .attr("d", line)
}
```

В результате при клике на кнопку **Следующий шаг** будет показан переход графика из одного состояния в следующее.

Самостоятельные задания

1. Реализовать динамическое построение графика при **t=0** при клике по кнопке **Начать**.
2. Реализовать анимацию графика от текущего шага до последнего при клике по кнопке **Анимировать**.

Пояснение. Анимацию от текущего шага до последнего можно реализовать с помощью вызова рекурсивной функции по окончании анимации текущего шага:

```
let animate = () => {

      // переход к следующему шагу

      // условие выхода из функции

    svg
      // определение данных и анимация очередного шага
      …
        .on("end", animate);
};
```