Отчет по кубической сплайн-интерполяции на Node.js

1. Чтение данных из файла

Данные, представляющие координаты точек (x_i, y_i), читаются из файла `input.txt` и сохраняются в виде массива объектов с ключами `x` и `y`.

```
// Чтение входных данных из файла input.txt
function readInput(filename) {
  const data = fs.readFileSync(filename, 'utf-8');
  const lines = data.trim().split('\n');
  const points = lines.map(line => {
    const [x, y] = line.trim().split(' ').map(Number);
    return { x, y };
  });
  return points;
}
```

2. Решение системы линейных уравнений методом Гаусса

Метод Гаусса используется для решения системы линейных уравнений, возникающей при вычислении коэффициентов сплайнов. Сначала проводится прямой ход метода Гаусса, а затем — обратный ход для нахождения решений.

Прямая и обратная фазы метода можно описать следующим образом:

- Прямая фаза: Преобразование матрицы системы к верхнетреугольному виду.
- Обратная фаза: Нахождение решений через подстановку.

```
// Решение системы линейных уравнений методом Гаусса
function solveGauss(matrix, vector) {
  const n = vector.length;
  for (let i = 0; i < n; i++) {
    // Прямой ход метода Гаусса
    for (let j = i + 1; j < n; j++) {
      const factor = matrix[j][i] / matrix[i][i];
      for (let k = i; k < n; k++) {
        matrix[j][k] -= factor * matrix[i][k];
      }
      vector[j] -= factor * vector[i];
    }
  }

  // Обратный ход метода Гаусса
  const result = new Array(n);
  for (let i = n - 1; i >= 0; i--) {
    result[i] = vector[i] / matrix[i][i];
    for (let j = i - 1; j >= 0; j--) {
      vector[j] -= matrix[j][i] * result[i];
    }
  }

  return result;
}
```

3. Вычисление коэффициентов кубических сплайнов

Для интерполяции кубическими сплайнами необходимо вычислить набор коэффициентов a_i, b_i, c_i, и d_i для каждого отрезка между соседними точками.

Полином для каждого отрезка выглядит так:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Коэффициенты вычисляются из системы уравнений с использованием условий непрерывности и гладкости сплайнов на границах отрезков.

```
// Вычисление кубических сплайнов
function computeCubicSpline(points) {
  const n = points.length - 1;
  const h = new Array(n);
  const alpha = new Array(n);

  // Формула для вычисления шага h_i: h_i = x_(i+1) - x_i
  for (let i = 0; i < n; i++) {
    h[i] = points[i + 1].x - points[i].x;
    // Формула для вычисления alpha_i: alpha_i = 3/h_i * (f_(i+1) - f_i)
    alpha[i] = (3 / h[i]) * (points[i + 1].y - points[i].y);
  }

  // Матрицы для решения методом Гаусса
  const matrix = new Array(n - 1).fill(0).map(() => new Array(n - 1).fill(0));
  const b = new Array(n).fill(0);
  const d = new Array(n).fill(0);
  const c = new Array(n + 1).fill(0);

  // Формируем матрицу для решения системы
  // Формула для диагональных элементов: 2(h_(i-1) + h_i)
  // Формула для элементов над и под диагональю: h_i
  for (let i = 1; i < n; i++) {
    matrix[i - 1][i - 1] = 2 * (h[i - 1] + h[i]);
    if (i > 1) matrix[i - 1][i - 2] = h[i - 1];
    if (i < n - 1) matrix[i - 1][i] = h[i];
  }

  const rhs = new Array(n - 1);
  // Формула для правой части системы: 3((f_(i+1) - f_i)/h_i - (f_i - f_(i-1))/h_(i-1))
  for (let i = 1; i < n; i++) {
    rhs[i - 1] = 3 * ((points[i + 1].y - points[i].y) / h[i] - (points[i].y - points[i - 1].y) / h[i - 1]);
  }

  // Решаем систему уравнений
  const cReduced = solveGauss(matrix, rhs);
```

```
// Восстанавливаем полный массив коэффициентов c
for (let i = 1; i < n; i++) {
  c[i] = cReduced[i - 1];
}

// Вычисляем остальные коэффициенты
// Формула для b_i: b_i = (f_(i+1) - f_i)/h_i - h_i(c_(i+1) + 2c_i)/3
// Формула для d_i: d_i = (c_(i+1) - c_i)/(3h_i)
for (let i = 0; i < n; i++) {
  b[i] = (points[i + 1].y - points[i].y) / h[i] - h[i] * (c[i + 1] + 2 * c[i]) / 3;
  d[i] = (c[i + 1] - c[i]) / (3 * h[i]);
}

return { a: points.map(p => p.y), b, c, d };
```