Códigos

- main.js

```
const io = require('console-read-write');
const fs = require('fs');
const util = require('util');
const createOutputFile = require('./utils/createOutputFile');
const Interpolation = require('./methods/Interpolation');
const Regression = require('./methods/Regression');

util.inspect.defaultOptions.depth = null;

async function main() {
  console.log('Digite o nome do arquivo:(ele deve estar na mesma pasta do executavel)');
  const fileName = await io.read();
  // const fileName = 'input2.txt';

  let buffer;
  try {
    buffer = fs.readFileSync(`./${fileName}`);
  } catch (error) {
    console.log('Arquivo não encontrado!');
    return;
  }

  const bufferAsString = buffer.toString();

  let [
    ICOD,
    n,
    ...rest
  ] = bufferAsString.split('\n');

  ICOD = parseInt(ICOD);
  n = parseInt(n);

  const pairs = rest.slice(0, n).map((line) => {
    const [x,y] = line.split(' ');
    return { x: parseFloat(x), y: parseFloat(y) };
  });
  const mainX = parseFloat(rest[n]);

  const params = {
    n,
    mainX,
    pairs,
  };

  console.log({ params });

  let answer;
  let method;

  switch(ICOD) {
    case 1:
      method = 'Interpolação';
      answer = Interpolation(params);
      break;
    case 2:
      method = 'Regressão';
      answer = Regression(params);
      break;
    default:
      break;
  }

  createOutputFile(answer, method);
}

main().then(response => console.log(response));
```

- utils/createOutputFile.js

```
const fs = require('fs');
```

```
function createOutputFile(answer, method) {
  let fileString = `Resolvido pelo método ${method}\n`;

  const {
    y,
    errors,
  } = answer || {};

  fileString += `y: ${y}\n`;

  if (errors && errors.length) {
    fileString += 'Erros:\n';
    errors.forEach((error) => {
      fileString += `${error}\n`;
    });
  }

  fileString += 'Ressalta-se que existe um erro associado a esses valores, pois são aproximações.\n';

  console.log({ file: fileString });
  fs.writeFileSync('answer.txt', fileString);
}

module.exports = createOutputFile;
```

- methods/Interpolation.js

```
function Interpolation({ n, mainX, pairs }) {
  let total = 0;
  for (const { x: currentX, y: currentY } of pairs) {
    let upSum = 1;
    let downSum = 1;
    for (const { x } of pairs) {
      if (x === currentX) continue;
      upSum *= (mainX - x);
      downSum *= (currentX - x);
    }

    total += currentY*(upSum/downSum);
  }

  console.log({ total });

  return {
    y: total,
  };
}

module.exports = Interpolation;
```

- methods/Regression.js

```
function Regression({ n, mainX, pairs }) {
  let sumx = 0;
  let sumy = 0;
  let sumxy = 0;
  let sumx2 = 0;

  for (const { x, y } of pairs) {
    sumx += x;
    sumy += y;
    sumxy += x*y;
    sumx2 += x**2;
  }

  const alfa = (n*sumxy - sumx*sumy)/(n*sumx2 - sumx**2);

  const beta = sumy/n - alfa*sumx/n;

  const y = alfa*mainX + beta;

  console.log({ alfa, beta });

  return {
    y,
  };
}
```

```
module.exports = Regression;
```