// Calcula os valores de alfa e wat

// Entradas: valores de pKa (array com 8 elementos (floating) faixa -2 a 16)

//           carga inicial (inteiro - faixa +4 a -8)

//           pH inicial (floating - faixa -2 a 16)

//           números de pH (inteiro - faixa 10 - 3000 pontos)

//           incremento pH (floating - 0,001 a 10)

//

// Array de saída de dados

//

//     col1     col2   col3 ... col9

//      pH0    alfa0  alfa1 ... alfa8

//      pH1    alfa0  alfa1 ... alfa8

//       .

//       .

//       .

//      pH (numero total de pontos)

//

//

let alpha = {

  valor: [],

  pH: [],

};

let temp = {

  expo: [],

  pKas\_soma: [],

  dif\_vector: [],

  somatorio: 0,

  alfa: [],

};

function calculateAlpha() {

  temp.pKas\_soma[0] = 0;

  pKa = [];

  i = 1;

  while (i < 8) {

    // preenche as colunas de pKa com valores do sistema Selecionado

    each\_pKa = document.getElementById(i.toString()).innerHTML;

    pKa.push(each\_pKa);

    i = i + 1;

  }

  console.log(

    "+++++ 1 +++++",

    temp.alfa,

    "alfa0",

    temp.alfa[0],

    "i",

    alpha.pH[i],

    "valor",

    alpha.valor

  );

  // NÚMERO DE PONTOS PARA SEREM CALCULADOS

  num\_point = 5;

  //new Array(num\_point);

  a = 10;

  // calcula os somatorios de pKa

  for (i = 1; i < 10; i++) {

    temp.pKas\_soma[i] = parseInt(pKa[i - 1]) + temp.pKas\_soma[i - 1];

  }

  somatorio = 0;

  //

  // realiza o cálculo dos alfas

  // para cada pH de pHvector,

  //

  //

  //alpha.pH.forEach((number\_pH, index\_pH) => {

  for (i = 0; i < num\_point; i++) {

    // cria vetor de pH com num\_point valores

    intervalo = 14 / num\_point;

    j = i \* intervalo;

    alpha.pH[i] = j;

    //

    // temp.dif\_vector = i\*pH-soma(pKa[até i])

    //

    for (index\_pKa = 0; index\_pKa < 8; index\_pKa++) {

      temp.dif\_vector[index\_pKa] =

        index\_pKa \* alpha.pH[i] - temp.pKas\_soma[index\_pKa];

    }

    //

    //temp.expo = 10^(temp.dif\_vector)

    // somatorio = soma (temp.expo)

    //           = 1/(1+ soma(10^(dif\_vetor)))

    //

    temp.expo = math.map(temp.dif\_vector, math.exp);

    temp.expo.forEach((*number\_termo*, *index\_termo*) => {

      temp.somatorio += temp.expo[index\_termo];

    });

    //

    // alfas temporários:

    // alfa0 = 1/somatorio

    //

    temp.alfa[0] = 1 / temp.somatorio;

    // reseta o somatorio

    temp.somatorio = 0;

    //

    // calcula os outros alfas:   alfa(i) = alfa(0)\* 10^(dif\_vector(i))

    //

    console.log("+++++ 1 +++++", temp.alfa);

    for (index\_pKa = 0; index\_pKa < 8; index\_pKa++) {

      temp.alfa[i] = temp.expo[index\_pKa] \* temp.alfa[0];

      console.log("+++++ 2 +++++", temp.alfa);

    }

    //

    //cria um array na linha seguinte com os valores de alfa

    //

    alpha.valor.push(temp.alfa);

    console.log("------------ alfas --------", alpha.valor);

    //;

    //console.log("+++++ 2 +++++", temp.alfa);

    //console.log("+++++ 3 +++++", temp.alfa);

    //DDE\_1.data.datasets[0].data = alpha0;

    //DDE\_1.data.datasets[0].data);

    //DDE\_1.update;

    //return; //alpha0;

  }

  console.log("\*\*\*\*\*fim\*\*\*\*\* alpha", alpha.valor, alpha.pH);

}

/\* grafico

let DDE\_fora = document.getElementById("DDE").getContext("2d");

dataX = [1, 2, 3, 4, 5];

dataY = [2, 3, 5, 9, 10];

let DDE\_out = new Chart(DDE\_fora, {

  type: "bar",

  data: {

    labels: alpha.pH,

    datasets: [{ data: alpha0, showLine: true }],

  },

});

\*/

let somaExpo = expo.reduce((*accumulator*, *currentValue*, *currentIndex*) => {

      temp\_phValues = 0;

      if (currentIndex === 0) {

        expo = Math.pow(

          10,

          currentIndex \* temp\_phValues - pkaSoma[currentIndex]

        );

        accumulator.push(currentValue);

      } else {

        accumulator.push(accumulator[currentIndex - 1] + currentValue);

      }

      return accumulator;

    }, []);

    console.log(somaExpo, expo);

  }

function calculateAlpha(*systemParameters*) {

  // Calculate cumulative sum array for pKa values

  let pkaSoma = systemParameters.pKaArray.reduce(

    (*accumulator*, *currentValue*) => {

      if (accumulator.length > 0) {

        accumulator.push(accumulator[accumulator.length - 1] + currentValue);

      } else {

        accumulator.push(currentValue);

      }

      return accumulator;

    },

    []

  );

  let phValues = [];

  for (let i = 0; i < systemParameters.num\_point; i++) {

    phValues.push(systemParameters.phInicial + i \* systemParameters.incremento);

    //

    systemParameters.pKaArray.forEach((*value*, *index*) => {

      expo = Math.pow(10, index \* phValues - pkaSoma[index]);

      if (index === 0) {

        expoSoma = expo;

      } else {

        expoSoma = parseFloat(expo) + parseFloat(expoSoma);

        console.log("soma", expoSoma, "expo", expo);

      }

    });

  }

  let expoValues = phValues.map((*phAtual*) => {

    return systemParameters.pKaArray.map((*pKa*, *index*) => {

      let pkaCumulativeSum = pkaSoma[index];

      return Math.pow(10, pKa \* phAtual - pkaCumulativeSum);

    });

  });

}