Univerzitet u Beogradu
Elektrotehnički fakultet

Odabrana poglavlja iz numeričke analize

Prvi projektni zadatak:

Najbolje racionalne aproksimacije realnih brojeva

Student:
Miloš Stojanović 2022/3175

Beograd, školska 2023/2024

Sadržaj

Projektni zadatak

Najbolje racionalne aproksimacije I i II vrste. *Neka je dat realan broj α. Racionalni broj p/q je najbolja racionalna aproksimacija realnog broja I vrste ako važi nejednakost*

$$\left|\alpha - \frac{p}{q}\right| < \left|\alpha - \frac{r}{s}\right|$$

za sve razlomke r/s ≠ p/q takve da 0 < s ≤ q. Racionalni broj p/q je najbolja racionalna aproksimacija II vrste ako važi nejednakost

$$|q\alpha - p| < |s\alpha - r|$$

za sve razlomke r/s ≠ p/q takve da 0 < s ≤ q.

Postavka projektnog zadatka:

Neka je dat pozitivan realan broj α sa konačnim decimalskim zapisom i neka su dati prirodni brojevi n i m, tako da n < m. Formirati niz razlomaka p/q takvih da za imenilac q važi n ≤ q ≤ m (tj. q = n, n + 1, . . . , m) i pri tom imeniocu q pridružujemo brojilac p koji određujemo zaokruživanjem na najbliži priodan broj proizvoda α · q. Predstaviti svaki razlomak p/q u obliku verižnog razlomka. U nizu razlomaka p/q izdvojiti:

- *najbolje racionalne aproksimacije I vrste,*
- *najbolje racionalne aproksimacije II vrste,*
- *sortirati sve razlomke p/q po uslovu minimalnosti apsolutne greške* $\left|\alpha - \frac{p}{q}\right|$.

Rešenje problema

Za rešavanje ovog problema korišćen je programski jezik JavaScript u okviru React biblioteke, kako bi se omogućio interaktivniji rad programa. Korišćene su isključivo predefinisane funkcionalnosti koje pružaju JavaScript i React tako da nije korišćena nijedna dodatna biblioteka.

U okviru aplikacije definisana je klasa Fraction koja predstavlja apstrakciju razlomka kako bismo lakše organizovali program kao logičku celinu. Klasa sadrži sledeća polja:

- p -- brojilac razlomka
- q -- imenilac razlomka
- coefficients -- niz verižnih decimala posmatranog razlomka p/q
- errorValue -- predstavlja vrednost apsolutne greške $\left|\alpha - \frac{p}{q}\right|$
- isFirst -- informacija da li je razlomak najbolja racionalna aproksimacija I vrste
- isSecond -- informacija da li je razlomak najbolja racionalna aproksimacija II vrste

Pri kreiranju objekta klase Fraction, konstruktoru se prosleđuju vrednosti p, q i α (pozitivan realan broj čiju racionalnu aproksimaciju određujemo). Nakon dodeljivanja vrednosti ovim poljima unutar objekta, najpre se izračunava vrednost apsolutne greške (po formuli $\left|\alpha - \frac{p}{q}\right|$), a potom se poziva i funkcija calculateContinuedFractionRepresentation (pomoću koje se izračunavaju verižne decimale tekućeg razlomka).

```
class Fraction {
  constructor(p, q, alfa) {
    this.p = p;
    this.q = q;
    this.alfa = alfa
    this.coefficients = [];
    this.isFirst = false
    this.isSecond = false
    this.errorValue = Math.abs(this.alfa - this.p/this.q)
    this.calculateContinuedFractionRepresentation()
  }
```

Slika 1. Klasa Fraction sa svojim poljima i konstruktorom

Funkcija calculateContinuedFractionRepresentation koristi algoritam za računanje verižnih decimala i tako izračunate decimale smešta u polje klase – coefficients

```
calculateContinuedFractionRepresentation() { // postupak racunanja veriznih decimala
  let x = this.p / this.q;
  let a = Math.floor(x);
  let d = x - a;

  this.coefficients.push(a);
  while(d > 1e-8){

    x = 1 / d;
    a = Math.floor(x);
    d = x - a;
    this.coefficients.push(a);
  }

  // ukoliko je poslednja verizna decimala 1, izbaci je i prethodnu veriznu decimalu povecaj za 1
  if (this.coefficients[this.coefficients.length - 1] == 1){
    this.coefficients.pop();
    this.coefficients[this.coefficients.length - 1] += 1;
  }
}
```

Slika 2. Funkcija calculateContinuedFractionRepresentation

Pored navedenih polja i funkcija, klasa Fraction sadrži i pomoćnu metodu za ispis verižnih decimala razlomka u valjanom formatu – displayContinuedFractionRepresentation

```
displayContinuedFractionRepresentation() {
  // formatiraj verizne decimale za pravilan ispis
  let coefficientsDisplay = "";
  for (let i = 0; i < this.coefficients.length; i++){
    coefficientsDisplay += this.coefficients[i];
    if (i == 0) {
      coefficientsDisplay += "; "
    }
    else {
      coefficientsDisplay += ", "
    }
  }
  coefficientsDisplay = coefficientsDisplay.slice(0, -2)

  return "[" + coefficientsDisplay + "]";
}
```

Slika 3. Funkcija displayContinuedFractionRepresentation

Posle klase Fraction imamo definisane funkcije koje predstavljaju glavno telo programa. Najpre imamo funkcije calculateErrTypeI i calculateErrTypeII kojima se računa apsolutna greška potrebna za proveru da li je posmatrani razlomak najbolja racionalna aproksimacija I (calculateErrTypeI), odnosno II vrste (calculateErrTypeII).

```
function calculateErrTypeI(alfa, p, q){
  return Math.abs(alfa - p / q)
}

function calculateErrTypeII(alfa, p, q){
  return Math.abs(alfa * q - p)
}
```

Slika 4. Funkcije calculateErrTypeI i calculateErrTypeII

Funkcija determineIfFirstOrSecond za dati realan broj alfa, imenilac q i niz razlomaka fractions, proverava da li svaki od razlomaka iz niza fractions predstavlja najbolju racionalnu aproksimaciju I odnosno II vrste realnog broja alfa. U funkciji se najpre određuje minimalna greška za prvu i drugu vrstu počevši od imenioca 1 dok se ne stigne do q. Minimalna greška prve vrste smešta se u promenljivu minErrTypeI, dok se minimalna greška druge vrste smešta u promenljivu minErrTypeII.

```
const determineIfFirstOrSecond = (alfa, q, fractions) => {
  let minErrTypeI = 1e12;
  let minErrTypeII = 1e12;

  for(let s = 1; s < q; s++){
    // I
    let r = Math.round(alfa * s)
    if (s == 1){
      minErrTypeI = calculateErrTypeI(alfa, r, s)
    }
    let currentErr = calculateErrTypeI(alfa, r, s)
    if (currentErr < minErrTypeI){
      minErrTypeI = currentErr
    }

    // II
    if (s == 1){
      minErrTypeII = calculateErrTypeII(alfa, r, s)
    }
    currentErr = calculateErrTypeII(alfa, r, s)
    if (currentErr < minErrTypeII){
      minErrTypeII = currentErr
    }
  }
```

Slika 5. funkcija determineIfFirstOrSecond – određivanje minimalne greške za I i II vrstu

Potom se za svaki razlomak iz niza fractions proveravaju sledeci uslovi: najpre da li je zadovoljen uslov currentErr < minErrTypeII (ukoliko je to tačno, razlomak predstavlja najbolju racionalnu aproksimaciju II vrste, a samim tim i najbolju racionalnu aproksimaciju I vrste – vrednosti isSecond i isFirst se postavljaju na true; ažurira se vrednost minErrTypeII na trenutno izračunatu vrednost greške za II vrstu i potom se izračunava greška za I vrstu i ukoliko je i ona manja od minErrTypeI, onda se ažurira i vrednost za I vrstu).

Ukoliko nije zadovoljen uslov currentErr < minErrTypeII, onda se postavlja false za vrednost isSecond posmatranog razlomka jer on ne predstavlja najbolju racionalnu aproksimaciju II vrste. Potom se proverava da li posmatrani razlomak predstavlja najblju racinalnu aproksimaciju I vrste – računa se greška za I vrstu i proverava uslov currentErr < minErrTypeI. Ako je uslov ispunjen, razlomak zaista predstavlja najbolju racionalnu aproksimaciju I vrste i za posmatrani razlomak se postavlja firstMin na true i ažurira se vrednost minErrTypeI na currentErr, a ako uslov nije ispunjen onda se vrednost isFirst posmatranog razlomka postavlja na false.

```
for (let i = 0; i < fractions.length; i++) {
  let currentErr = calculateErrTypeII(alfa, fractions[i].p, fractions[i].q);

  if (currentErr < minErrTypeII){
    minErrTypeII = currentErr
    fractions[i].isSecond = true;
    fractions[i].isFirst = true;
    currentErr = calculateErrTypeI(alfa, fractions[i].p, fractions[i].q);
    if (currentErr < minErrTypeI){
      minErrTypeI = currentErr;
    }
  }

  else{
    fractions[i].isSecond = false;
    currentErr = calculateErrTypeI(alfa, fractions[i].p, fractions[i].q);
    if (currentErr < minErrTypeI) {
      minErrTypeI = currentErr;
      fractions[i].isFirst = true;
    }
    else {
      fractions[i].isFirst = false;
    }
  }
}
```

Slika 6. funkcija determineIfFirstOrSecond – iteracija kroz sve razlomke i određivanje da li predstavljaju najbolju racionalnu aproksimaciju I/II vrste

Od ostalih funkcija koje su korišćene za realizaciju projekta, značajne su sledeće funkcije:

- handleInput: funkcija koja proverava da li su u formi aplikacije unete ispravne vrednosti za parametre α, N i M (sve vrednosti se citaju direktno iz forme)

```
// funkcija za regulisanje pravilnog unosa alfa, N i M parametara
const handleInput = () => {

  if (isNaN(alfa)) {
    alert("Za broj alfa nije uneta ispravna brojčana vrednost!")
    return;
  }
  if (isNaN(N) || N <= 0) {
    alert("Za broj N nije uneta ispravna brojčana vrednost!")
    return false;
  }
  if (isNaN(M)) {
    alert("Za broj M nije uneta ispravna brojčana vrednost!")
    return false;
  }
  if (M <= N) {
    alert("Nisu unete ispravne brojčane vrednosti, mora da važi M >= N!")
    return false;
  }

  return true;
}
```

Slika 7. funkcija handleInput

- handleButtonClick: poziva se klikom na dugme „Izračunaj“ – najpre proverava da li su uneti ispravni podaci putem forme, potom formira dinamicki niz razlomaka i nad njima prvo poziva funkciju determineIfFirstOrSecond i potom ih sortira po uslovu minimalnosti apsolutne greške. Na kraju konačnu vrednost postavlja u pomoćnu promenljivu, koja se ispisuje u tabeli ispod glavne forme

```
// Funkcija koja se poziva klikom na dugme "Izracunaj"
const handleButtonClick = () => {

  // provera da li su uneti ispravni podaci
  let inputValid = handleInput();
  if(!inputValid){
    return;
  }
  setInputDisabled(true);

  // Formiranje dinamickog niza razlomaka
  let fractions = []
  for(let i = N; i <= M; i++){
    if (gcd(Math.round(  alfa * i), i) != 1){
      continue; // Neredukovani razlomak, nama je visak
    }
    let fraction = new Fraction(Math.round( alfa * i), i, alfa);
    fractions.push(fraction)
  }
  // proveri za sve razlomke da li su najbolja racionalna aproksimacija I/II vrste
  determineIfFirstOrSecond(alfa, N, fractions)
  // sortiramo razlomke po uslovu minimalnosti apsolutne greske
  fractions.sort((a, b) => Math.abs(a.errorValue) - Math.abs(b.errorValue));

  M1 = M;
  N1 = N;
  // postavljamo sortirane razlomke za ispis u tabeli
  setData(fractions)
};
```

Slika 8. funkcija handleButtonClick

- komponenta MyTable: prima podatke koji su rezultat rada funkcije handleButtonClick i vrši njihov prikaz ispod forme u kojoj se unose podaci za rad programa

```
// komponenta koja sluzi za ispis rezultata racunanja
const MyTable = ({ data }) => {

  // slucaj da jos uvek nije izracunat rezultat programa
  if (!data || !Array.isArray(data)) {
    return <div></div>;
  }

  // filtriranje najboljih racionalnih aproksimacija I i II vrste
  let fractionsTypeII = [];
  let fractionsTypeI = [];
  for (let i = 0; i < data.length; i++){

    if (data[i].isSecond){
      fractionsTypeII.push(data[i]);
    }

    if (data[i].isFirst){
      fractionsTypeI.push(data[i]);
    }

  }
```

Slika 9. komponenta MyTable – filtriranje najboljih racionalnih aproksimacija I/II vrste za prikaz

```
// sortiranje racionalnih aproksimacija rastuce po vrednosti imenionca
fractionsTypeI.sort((a, b) => a.q - b.q);
fractionsTypeII.sort((a, b) => a.q - b.q);


// formiranje stringa za sortirani ispis razlomaka koji predstavljaju najbolje racionalne aproksimacije
let fractionsII = "";
for (let i = 0; i < fractionsTypeII.length; i++){
  fractionsII = fractionsII + `${fractionsTypeII[i].p}/${fractionsTypeII[i].q}` + ", "
}
fractionsII = fractionsII.slice(0, -2);

let fractionsI = "";
for (let i = 0; i < fractionsTypeI.length; i++){
  fractionsI = fractionsI + `${fractionsTypeI[i].p}/${fractionsTypeI[i].q}` + ", "
}
fractionsI = fractionsI.slice(0, -2);
```

Slika 10. komponenta MyTable – formiranje prikaza najboljih racionalnih aproksimacija I/II vrste

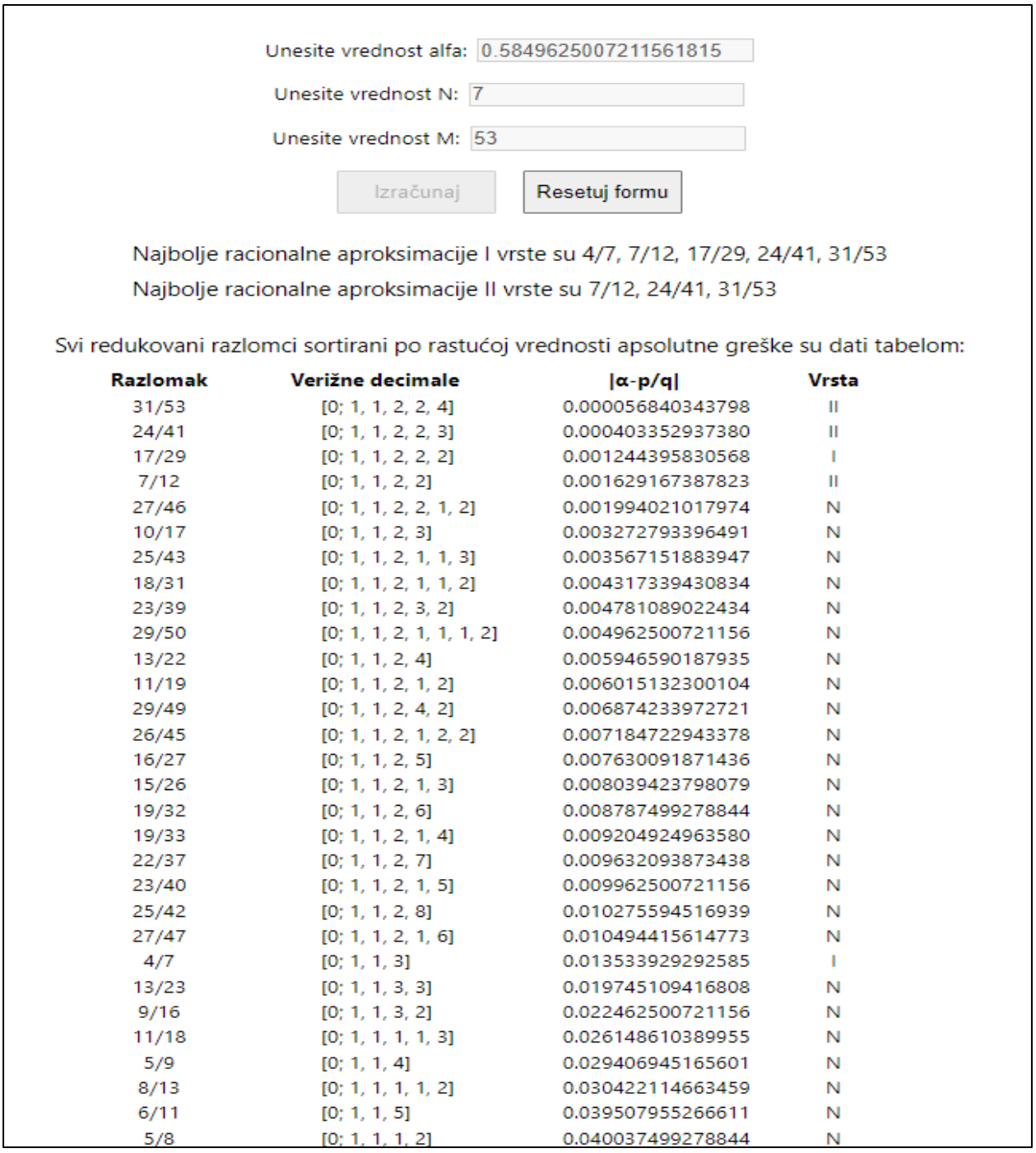
```
return (
  <>
    <div>
      <div style={{marginTop: 30, marginBottom: 10, fontSize: 22}}>Najbolje racionalne aproksimacije I vrste su {fractionsI}</div>
      <div style={{marginBottom: 15, fontSize: 22, textAlign: "left"}}>Najbolje racionalne aproksimacije II vrste su {fractionsII}</div>
    </div>
    <div style={{marginTop: 20, marginBottom: 10, fontSize: 22}}>Svi redukovani razlomci sortirani po rastućoj vrednosti apsolutne greške su dati tabelom:</div>
    <table style={{fontSize: 20}}>
      {/* Table header */}
      <thead>
        <tr>
          <th style={{width: 100}}>Razlomak</th>
          <th style={{width: 300}}>Verižne decimale</th>
          <th style={{width: 200}}>|α-p/q|</th>
          <th style={{width: 150}}>Vrsta</th>
        </tr>
      </thead>
      {/* Table body */}
      <tbody>
        {data.map((item, index) => (
          <tr key={index}>
            <td style={{textAlign: "center"}}>{`${item.p}/${item.q}`}</td>
            <td style={{textAlign: "left", paddingLeft: 100}}>{item.displayDecimals()}</td>
            <td style={{textAlign: "right"}}>{item.errorValue.toFixed(15)}</td>
            <td style={{textAlign: "center"}}>{item.isSecond ? "II" : item.isFirst ? "I" : "N"}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </>
);
```

Slika 11. komponenta MyTable – tabela sa prikazom svih generisanih razlomaka, sortiranih po minimalnosti apsolutne greške odstupanja, sa posebnom naznakom koji razlomci su najbolje racionalne aproksimacije I/II vrste a koji ne

Primeri za testiranje

Primer 1

$\alpha = 0.5849625007211561815$, N = 7, M = 53



Razlomak	Verižne decimale	|α-p/q|	Vrsta
31/53	[0; 1, 1, 2, 2, 4]	0.000056840343798	II
24/41	[0; 1, 1, 2, 2, 3]	0.000403352937380	II
17/29	[0; 1, 1, 2, 2, 2]	0.001244395830568	I
7/12	[0; 1, 1, 2, 2]	0.001629167387823	II
27/46	[0; 1, 1, 2, 2, 1, 2]	0.001994021017974	N
10/17	[0; 1, 1, 2, 3]	0.003272793396491	N
25/43	[0; 1, 1, 2, 1, 1, 3]	0.003567151883947	N
18/31	[0; 1, 1, 2, 1, 1, 2]	0.004317339430834	N
23/39	[0; 1, 1, 2, 3, 2]	0.004781089022434	N
29/50	[0; 1, 1, 2, 1, 1, 1, 2]	0.004962500721156	N
13/22	[0; 1, 1, 2, 4]	0.005946590187935	N
11/19	[0; 1, 1, 2, 1, 2]	0.006015132300104	N
29/49	[0; 1, 1, 2, 4, 2]	0.006874233972721	N
26/45	[0; 1, 1, 2, 1, 2, 2]	0.007184722943378	N
16/27	[0; 1, 1, 2, 5]	0.007630091871436	N
15/26	[0; 1, 1, 2, 1, 3]	0.008039423798079	N
19/32	[0; 1, 1, 2, 6]	0.008787499278844	N
19/33	[0; 1, 1, 2, 1, 4]	0.009204924963580	N
22/37	[0; 1, 1, 2, 7]	0.009632093873438	N
23/40	[0; 1, 1, 2, 1, 5]	0.009962500721156	N
25/42	[0; 1, 1, 2, 8]	0.010275594516939	N
27/47	[0; 1, 1, 2, 1, 6]	0.010494415614773	N
4/7	[0; 1, 1, 3]	0.013533929292585	I
13/23	[0; 1, 1, 3, 3]	0.019745109416808	N
9/16	[0; 1, 1, 3, 2]	0.022462500721156	N
11/18	[0; 1, 1, 1, 1, 3]	0.026148610389955	N
5/9	[0; 1, 1, 4]	0.029406945165601	N
8/13	[0; 1, 1, 1, 1, 2]	0.030422114663459	N
6/11	[0; 1, 1, 5]	0.039507955266611	N
5/8	[0; 1, 1, 1, 2]	0.040037499278844	N

Slika 12. Ispis programa za vrednosti $\alpha = 0.5849625007211561815$, N=7 i M = 53

Primer 2

α = 3.141258745821457

N = 1

M = 25

Razlomak	Verižne decimale	|α-p/q|	Vrsta
22/7	[3; 7]	0.001598397035686	II
69/22	[3; 7, 3]	0.004895109457821	N
47/15	[3; 7, 2]	0.007925412488124	N
63/20	[3; 6, 1, 2]	0.008741254178543	N
72/23	[3; 7, 1, 2]	0.010823963212761	N
41/13	[3; 6, 2]	0.012587408024697	N
25/8	[3; 8]	0.016258745821457	N
60/19	[3; 6, 3]	0.016635991020648	N
79/25	[3; 6, 4]	0.018741254178543	N
53/17	[3; 8, 2]	0.023611686997928	N
19/6	[3; 6]	0.025407920845209	I
28/9	[3; 9]	0.030147634710346	N
35/11	[3; 5, 2]	0.040559435996725	N
31/10	[3; 10]	0.041258745821457	N
16/5	[3; 5]	0.058741254178543	I
13/4	[3; 4]	0.108741254178543	I
3/1	[3]	0.141258745821457	II

Slika 13. Ispis programa za vrednosti α = 3.141258745821457, N=1 i M = 25

Kod

Napisan u razvojnom okruženju Visual Studio Code, korišćenjem ES6 standarda.

```
import React, { useState } from 'react';

class Fraction {
  constructor(p, q, alfa) {
    this.p = p;
    this.q = q;
    this.alfa = alfa
    this.coefficients = [];
    this.isFirst = false
    this.isSecond = false
    this.errorValue = Math.abs(this.alfa - this.p/this.q)
    this.calculateContinuedFractionRepresentation()
  }

  calculateContinuedFractionRepresentation() { // postupak racunanja veriznih
decimala
    let x = this.p / this.q;
    let a = Math.floor(x);
    let d = x - a;

    this.coefficients.push(a);
    while(d > 1e-8){

      x = 1 / d;
      a = Math.floor(x);
      d = x - a;
      this.coefficients.push(a);
    }

    // ukoliko je poslednja verizna decimala 1, izbaci je i prethodnu veriznu
decimalu povecaj za 1
    if (this.coefficients[this.coefficients.length - 1] == 1){
      this.coefficients.pop();
      this.coefficients[this.coefficients.length - 1] += 1;
    }
  }

  displayContinuedFractionRepresentation() {
    // formatiraj verizne decimale za pravilan ispis
```

```
    let coefficientsDisplay = "";
    for (let i = 0; i < this.coefficients.length; i++){
      coefficientsDisplay += this.coefficients[i];
      if (i == 0) {
        coefficientsDisplay += "; "
      }
      else {
        coefficientsDisplay += ", "
      }
    }
    coefficientsDisplay = coefficientsDisplay.slice(0, -2)

    return "[" + coefficientsDisplay + "]";
  }

}


// komponenta koja sluzi za ispis rezultata racunanja
const MyTable = ({ data }) => {

  // slucaj da jos uvek nije izracunat rezultat programa
  if (!data || !Array.isArray(data)) {
    return <div></div>;
  }

  // filtriranje najboljih racionalnih aproksimacija I i II vrste
  let fractionsTypeII = [];
  let fractionsTypeI = [];
  for (let i = 0; i < data.length; i++){

    if (data[i].isSecond){
      fractionsTypeII.push(data[i]);
    }

    if (data[i].isFirst){
      fractionsTypeI.push(data[i]);
    }

  }

  // sortiranje racionalnih aproksimacija rastuce po vrednosti imenionca
  fractionsTypeI.sort((a, b) => a.q - b.q);
  fractionsTypeII.sort((a, b) => a.q - b.q);
```

```
  // formiranje stringa za sortirani ispis razlomaka koji predstavljaju najbolje
racionalne aproksimacije
  let fractionsII = "";
  for (let i = 0; i < fractionsTypeII.length; i++){
    fractionsII = fractionsII + `${fractionsTypeII[i].p}/${fractionsTypeII[i].q}`
+ ", "
  }
  fractionsII = fractionsII.slice(0, -2);

  let fractionsI = "";
  for (let i = 0; i < fractionsTypeI.length; i++){
    fractionsI = fractionsI + `${fractionsTypeI[i].p}/${fractionsTypeI[i].q}` +
", "
  }
  fractionsI = fractionsI.slice(0, -2);


  return (
    <>
      <div>
        <div style={{marginTop: 30, marginBottom: 10, fontSize: 22}}>Najbolje
racionalne aproksimacije I vrste su {fractionsI}</div>
        <div style={{marginBottom: 15, fontSize: 22, textAlign: "left"}}>Najbolje
racionalne aproksimacije II vrste su {fractionsII}</div>
      </div>
      <div style={{marginTop: 20, marginBottom: 10, fontSize: 22}}>Svi redukovani
razlomci sortirani po rastućoj vrednosti apsolutne greške su dati tabelom:</div>
      <table style={{fontSize: 20}}>
        {/* Table header */}
        <thead>
          <tr>
            <th style={{width: 100}}>Razlomak</th>
            <th style={{width: 300}}>Verižne decimale</th>
            <th style={{width: 200}}>|α-p/q|</th>
            <th style={{width: 150}}>Vrsta</th>
          </tr>
        </thead>
        {/* Table body */}
        <tbody>
          {data.map((item, index) => (
            <tr key={index}>
              <td style={{textAlign: "center"}}>{`${item.p}/${item.q}`}</td>
              <td style={{textAlign: "left", paddingLeft:
100}}>{item.displayContinuedFractionRepresentation()}</td>
              <td style={{textAlign: "right"}}>{item.errorValue.toFixed(15)}</td>
```

```
              <td style={{textAlign: "center"}}>{item.isSecond ? "II" :
item.isFirst ? "I" : "N"}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </>
  );
};

const App = () => {
  // State to store input values
  const [N, setN] = useState("")
  const [M, setM] = useState("")
  const [alfa, setAlfa] = useState("")
  const [inputDisabled, setInputDisabled] = useState(false);
  let M1, N1;
  const [data, setData] = useState(null);


  function tryParseN(val) {
    let n = parseInt(val, 10)

    if (isNaN(n) || n <= 0) {
      alert("Za broj N nije uneta ispravna brojčana vrednost!")
      return false;
    }
    setN(n)
  }


  function tryParseM(val) {
    let m = parseInt(val, 10)

    if (isNaN(m)) {
      alert("Za broj M nije uneta ispravna brojčana vrednost!")
      return false;
    }
    setM(m)
  }

  // funkcija za regulisanje pravilnog unosa alfa, N i M parametara
  const handleInput = () => {

    if (isNaN(alfa)) {
```

```
    alert("Za broj alfa nije uneta ispravna brojčana vrednost!")
    return;
  }
  if (isNaN(N) || N <= 0) {
    alert("Za broj N nije uneta ispravna brojčana vrednost!")
    return false;
  }
  if (isNaN(M)) {
    alert("Za broj M nije uneta ispravna brojčana vrednost!")
    return false;
  }
  if (M <= N) {
    alert("Nisu unete ispravne brojčane vrednosti, mora da važi M >= N!")
    return false;
  }

  return true;
}

function calculateErrTypeI(alfa, p, q){
  return Math.abs(alfa - p / q)
}

function calculateErrTypeII(alfa, p, q){
  return Math.abs(alfa * q - p)
}

const determineIfFirstOrSecond = (alfa, q, fractions) => {
  let minErrTypeI = 1e12;
  let minErrTypeII = 1e12;

  for(let s = 1; s < q; s++){
    // I
    let r = Math.round(alfa * s)
    if (s == 1){
      minErrTypeI = calculateErrTypeI(alfa, r, s)
    }
    let currentErr = calculateErrTypeI(alfa, r, s)
    if (currentErr < minErrTypeI){
      minErrTypeI = currentErr
    }

    // II
    if (s == 1){
      minErrTypeII = calculateErrTypeII(alfa, r, s)
```

```
    }
    currentErr = calculateErrTypeII(alfa, r, s)
    if (currentErr < minErrTypeII){
      minErrTypeII = currentErr
    }
  }

  for (let i = 0; i < fractions.length; i++) {
    let currentErr = calculateErrTypeII(alfa, fractions[i].p, fractions[i].q);

    if (currentErr < minErrTypeII){
      minErrTypeII = currentErr
      fractions[i].isSecond = true;
      fractions[i].isFirst = true;
      currentErr = calculateErrTypeI(alfa, fractions[i].p, fractions[i].q);
      if (currentErr < minErrTypeI){
        minErrTypeI = currentErr;
      }
    }

    else{
      fractions[i].isSecond = false;
      currentErr = calculateErrTypeI(alfa, fractions[i].p, fractions[i].q);
      if (currentErr < minErrTypeI) {
        minErrTypeI = currentErr;
        fractions[i].isFirst = true;
      }
      else {
        fractions[i].isFirst = false;
      }
    }
  }

}

// Funkcija koja se poziva klikom na dugme "Izracunaj"
const handleButtonClick = () => {

  // provera da li su uneti ispravni podaci
  let inputValid = handleInput();
  if(!inputValid){
    return;
  }
  setInputDisabled(true);
```

```
    // Formiranje dinamickog niza razlomaka
    let fractions = []
    for(let i = N; i <= M; i++){
      if (gcd(Math.round(  alfa * i), i) != 1){
        continue; // Neredukovani razlomak, nama je visak
      }
      let fraction = new Fraction(Math.round( alfa * i), i, alfa);
      fractions.push(fraction)
    }
    // proveri za sve razlomke da li su najbolja racionalna aproksimacija I/II
vrste
    determineIfFirstOrSecond(alfa, N, fractions)
    // sortiramo razlomke po uslovu minimalnosti apsolutne greske
    fractions.sort((a, b) => Math.abs(a.errorValue) - Math.abs(b.errorValue));

    M1 = M;
    N1 = N;
    // postavljamo sortirane razlomke za ispis u tabeli
    setData(fractions)
  };

  function gcd(a, b) {
    a = Math.abs(a);
    b = Math.abs(b);

    while (b !== 0) {
      const temp = b;
      b = a % b;
      a = temp;
    }

    return a;
  }


  return (
    <div style={{display: "flex", flexDirection: "column", alignItems: "center",
justifyContent: "center", marginTop: "20vh"}}>
      <label style={{fontSize: 20}}>
        Unesite vrednost alfa:
        <input
          style={{marginLeft: 10, fontSize: 20}}
          type="text"
          value={alfa}
          disabled={inputDisabled}
```

```
          onChange={(e) => { setAlfa(e.target.value);}}
        />
      </label>
      <br />
      <label style={{fontSize: 20}}>
        Unesite vrednost N:
        <input
          style={{marginLeft: 10, fontSize: 20}}
          type="text"
          value={N}
          disabled={inputDisabled}
          onChange={(e) => { tryParseN(e.target.value);}}
        />
      </label>
      <br />
      <label style={{fontSize: 20}}>
        Unesite vrednost M:
        <input
        style={{marginLeft: 10, fontSize: 20}}
          type="text"
          value={M}
          disabled={inputDisabled}
          onChange={(e) => { tryParseM(e.target.value);}}
        />
      </label>
      <br />
      <div style={{display: "flex", direction: "row"}}>
        <button
          style={{width: 150, height: 50, fontSize: 20, marginRight: 25}}
          disabled={inputDisabled}
          onClick={handleButtonClick}>Izračunaj
        </button>
        <button
        style={{width: 150, height: 50, fontSize: 20}}
        disabled={!inputDisabled}
        onClick={() => {window.location.reload()}}>Resetuj formu
      </button>
      </div>

      <MyTable data={data} />
    </div>
  );
};

export default App;
```