C# aplikacija za crtanje matematičkih funkcija i određivanje tijeka

Birtić, Dominik

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

C# APLIKACIJA ZA CRTANJE MATEMATIČKIH FUNKCIJA I ODREĐIVANJA TIJEKA

Završni rad

Dominik Birtić

Osijek, 2017.

SADRŽAJ

1. UVOD

Tema ovog završnog rada jeC# aplikacija za crtanje matematičkih funkcija i određivanje tijeka funkcije.Prvo se obrađuje teorija vezana za funkcije uz primjere. Zatim se obrađuje računanje tijeka funkcije. Naposlijetku se prikazuje aplikacija u C#-u koja crta funkcije i računa njihov tijek.

Glavni dio rada je podijeljen tri dijela: *funkcija, tijek funkcije i primjena u programskom jeziku C#.*

U poglavlju *Funkcija* su objašnjeni osnovni pojmovi funkcija. Dalje se opisuju ekstremi funkcija jer se oni koriste pri računanju tijeka funkcije. Te se na kraju opisuju ekstremi funkcije jednog argumenta, smjer konkavnosti i točke infleksije te asimptote.

U poglavlju *Tijek funkcije* se objašnjava način računanja tijeka funkcije uz riješen primjer na kojemu pokazujemo kako se svaki dio tijeka funkcije izračunava.

U poglavlju *Primjena u programskom jeziku C#* se prikazuje kod pomoću kojega se izračunava tijek funkcije te grafičko sučelje aplikacije na kojoj se nacrta unesena funkcija.

1.1 Zadatak završnog rada

Napraviti C# aplikaciju u kojoj korisnik unosi različite matematičke funkcije. Aplikacija treba nacrtati graf funkcije, izračunati derivacije, minimum i maksimum, intervale pada i rasta te odrediti tijek funkcije.

2. FUNKCIJA

2.1 Pojam funkcije

Funkcija ili preslikavanje je uređena trojka ($\boldsymbol{D}$, $\boldsymbol{K}$, $\boldsymbol{f}$) koja sadrži skupove $\boldsymbol{D}$, $\boldsymbol{K}$ i neko pravilo $\boldsymbol{f}$:$\boldsymbol{D} \rightarrow \boldsymbol{K}$ po kojem se svakome članu $\boldsymbol{x} \in \boldsymbol{D}$ pridržuje jedinstveni član $\boldsymbol{y} \in \boldsymbol{X}$ tako da je $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$. Skup $\boldsymbol{D}$ se naziva domena ili područje definicije funkcije $\boldsymbol{f}$, a skup $\boldsymbol{K}$ područje vrijednosti ili kodomena funkcije $\boldsymbol{f}$. Član domene $\boldsymbol{x}$ je nezavisna varijabla ili argument funckije $\boldsymbol{f}$, a član kodomene $\boldsymbol{y}$ je zavisna varijabla funkcije $\boldsymbol{f}$. [3]

Ako svakoj vrijednosti $\boldsymbol{x}$ koja pripada nekom skupu $\boldsymbol{E}$, odgovara jedna ili nekoliko vrijednosti varijable $\boldsymbol{y}$, tada $\boldsymbol{y}$ nazivamo *višeznačnom funkcijom* od $\boldsymbol{x}$, definiranom na skupu $\boldsymbol{E}$. [1, str. 11]

2.1.1 Područje definicije funkcije

Sve vrijednosti $\boldsymbol{x}$, za koje je zadana funkcija definirana nazivamo *područjem definicije* te funkcije. U jednostavnim slučajevima područje definicije funkcije jest: ili *odsječak* (*zatvoreni interval, segment*) *[a,b]*, tj. skup realnih brojeva $\boldsymbol{x}$, koji zadovoljavaju nejednadžbu$a \leq x \leq b$; ili *otvoreniinterval(a,b),* tj. skup realnih brojeva $\boldsymbol{x}$, koji zadovoljavaju nejednadžbu $a < x < b$. [1, str. 11]

2.1.2 Inverzne funkcije

Ako za sve vrijednosti zavrisne varijable $\boldsymbol{y}$ funkcije $\boldsymbol{f}(\boldsymbol{x})$ jednadžbu $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ možemo jednoznačno riješiti po varijabli $\boldsymbol{x}$, tj. ako postoji takva funkcija $x = g(y)$ da je $y \equiv f[g(y)]$, tada je funkcija $x = g(y)$, ili uobičajenom oznakom $y = g(x)$, inverzna od $y = f(x)$. Očigledno je da je $g[f(x)] \equiv x$, tj. funkcije $f(x)$ i $g(x)$ su međusobno inverzne.

Općenito, jednadžba $y = f(x)$ definira višeznačnu inverznu funkciju $x = f^{-1}(y)$ i to takvu, da je $y \equiv f(f^{-1}(y))$ za sve y koji su vrijednosti funkcije $f(x)$. [1, str. 11]

2.1.3 Složene i implicitne funkcije

Funckiju y od x zadanu lancem jednakosti $y = f(u)$, gdje je $u = \varphi(x)$ ($f(u)$ je definirano za sve u iz skupa vrijednosti funkcije $\varphi(x)$) itd. nazivamo *složenom* ili *funkcijom od funkcije*.

Funkciju, zadanu jednadžbom, koja nije riješena po zavisnoj varijabli, nazivamo *implicitnom*. Na primjer jednadžba $x^3 + y^3 = 1$ određuje y kao implicitnu funkciju od x.[1, str. 12]

2.1.4 Neprekidnost funkcije

Neku funkciju $f(x)$ nazivamo *neprekidnom* za nekakav $x = \xi$ (ili u točki ξ) ako je:

1) ta funkcija definirana u točki ξ, tj. postoji broj $f(\xi)$;

2) postoji konačni limes $\lim_{x \to \xi} f(x)$;

3) taj limes jednak vrijednosti funkcije u točki ξ, tj. $\lim_{x \to \xi} f(x) = f(\xi)$. [1, str. 36]

Ako je funkcija neprekinuta u svakoj točki nekog područja (intervala, raspona, segmenta itd.), onda se ona naziva *funkcija neprekinuta u tom području*.

2.2 Ekstremi funkcija

2.2.1 Ekstremi funkcije jednog argumenta

Ako postoji takva okolina točke x_0 da za svaku točku $x \neq x_0$ te okoline vrijedi nejednadžba $f(x) > f(x_0)$, tada točku x_0 nazivamo točkom *minimuma* funkcije $y = f(x)$ a broj $f(x_0)$*minimumom* funkcije $y = f(x)$. Također, ako za svaku točku $x \neq x_1$ neke okoline točke x_1 vrijedi nejednadžba $f(x) < f(x_1)$, tada x_1 nazivamo točkom *maksimuma* funkcije $f(x)$, a $f(x_1)$*maksimumom* funkcije. Točke *minimuma* ili *maksimuma* još nazivamo točkama *ekstrema*, a *minimum* ili *maksimum* funkcije *ekstremom* funkcije. [1, str. 84]

2.2.2 Smjer konkavnosti i točke infleksije

Za graf derivabilne funkcije $y = f(x)$ kažemo da je *konkavan nadolje* u intervalu (a, b) (*konkavan nagore* u intervalu (a_1, b_1)), ako postoje $a < x < b$ za koje se luk krivulje nalazi ispod (ili čak iznad ako postoje $a_1 < x < b_1$) tangente povučene u proizvoljno odabranoj točki intervala (a, b) (ili intervala (a_1, b_1)), osim točke dodira. Dovoljan je uvjet za konkavnost nadolje (nagore) grafa $y = f(x)$ da su zadovoljene nejednadžbe $f''(x) < 0$ ili $f''(x) > 0$ u odgovarajućem intervalu.

Točke infleksije su točke u kojima se mijenja smjer konkavnosti grafa funkcije, tj. za neku točku $(x_0,\ f(x_0))$ se mijenja smjer konkavnosti grafa $y = f(x_0)$. Za apscisu točke infleksije x_0 grafa funkcije $y = f(x)$ vrijedi $f''(x_0) = 0$ ili $f''(x_0)$ ne postoji. [1, str. 91]

2.2.3 Asimptote

Ako postoji neka točka (x, y) koja se neprekinuto pomiče po krivulji $y = f(x)$ tako da barem jedna koordinata točke teži k neizmjerenosti, a pri tome je udaljenost točke od nekog pravca teži nuli, tada se taj pravac naziva *asimptotom* krivulje. Postoje dvije vrste asimptota: vertikalne i kose asimptote. Vertikalne asimptote postoje ako postoji neki broj *a* za koji ako vrijedi:

$\lim_{x \to a} f(x) = \infty$, onda je pravac $x = a$ asimptota (*vertikalna asimptota*). Kose asimptote postoje ako postoje limesi: $\lim_{x \to +\infty} \frac{f(x)}{x} = k_1$ i $\log_{x \to +\infty}[f(x) - k_1 x] = b_1$, onda je pravac $y = k_1 x + b_1$ asimptota (*desna kosa* ili u slučaju kada je $k_1 = 0$, *desna horizontalna asimptota*). Ako postoje je limesi: $\lim_{x \to -\infty} \frac{f(x)}{x} = k_2$ i $\lim_{x \to -\infty} [f(x) - k_2 x] = b_2$, onda je pravac $y = k_2 x + b_2$ asimptota (*lijeva kosa* ili u slučaju $k_2 = 0$, *lijeva horizontalna asimptota*).

Graf funkcije $y = f(x)$ ne može imati više od jedne desne (kose ili horizontalne) niti više od jedne lijve (kose ili horizontalne) asimptote, s pretpostavkom da je funkcija jednoznačna. [1, str. 92, 93]

3. TIJEK FUNKCIJE

3.1 Rješavanje tijeka funkcije

Rješavanje tijeka funkcije je složen postupak u kojem se primjenjuje sve što je dosada navedeno o funkcijama. Rješavanje tijeka funkcije se sastoji iz idućih koraka: traženje domene funkcije, ispitivanje parnosti funkcije, ispitivanje periodičnosti funkcije, traženje nul-točki funkcije, traženje asimptota funkcije, traženje ekstrema funkcije, traženje intervala monotonosti funkcije, trženje intervala zakrivljenosti (konkavnost), traženje točka infleksije i crtanje grafa funkcije.

3.2 Primjer rješavanja tijeka funkcije

Kao primjer rješavanja funkcije ćemo koristiti ovu funkciju: $y = f(x) = \sqrt[3]{2x^2 - x^3}$. [2]

3.2.1 Traženje područja definicije

Domena funkcije je $\mathcal{D} = \mathbb{R}$.

3.2.2 Ispitivanje parnosti funkcije

Vrijedi $f(-1) = \sqrt[3]{2+1} = \sqrt[3]{3}$ dok je $f(1) = \sqrt[3]{2-1} = \sqrt[3]{1} = 1$. Zaključujemo da funkcija nije ni parna ni ne parna jer je $f(-x) \neq f(x)$ i $f(-x) \neq -f(x)$.

3.2.3 Ispitivanje periodičnosti funkcije

Funkcija nije periodična jer je funkcija elementarna funkcija i ne sadrži nijednu od trigonometrijskih funkcija.

3.2.4 Traženje nul-točki funkcije

Riješimo jednadžbu $y = 0$. Vrijedi $\sqrt[3]{2x^2 - x^3} = 0 \leftrightarrow 2x^2 - x^3 = 0 \leftrightarrow x^2(2-x) = 0$

pa su nul-točke jednake $x_1 = 0$ i $x_2 = 2$.

3.2.5 Traženje asimptota funkcije

Funkcija nema vertikalnih asimptota jer je$\mathcal{D} = \mathbb{R}$. Horizontalne asimptote: na lijevoj stranivrijedi $\lim_{x\to-\infty} f(x) = \lim_{x\to-\infty} \sqrt[3]{2x^2 - x^3} = \lim_{x\to-\infty} \sqrt[3]{x^3(\frac{2}{x} - 1)} = \lim_{x\to-\infty} x\sqrt[3]{\frac{2}{x} - 1} = (-\infty) \cdot (-1) = +\infty$

pa funkcija nema horizontalnu asimptotu u lijevoj strani. Na desnoj strani vrijedi

$$\lim_{x\to+\infty} f(x) = \lim_{x\to+\infty} \sqrt[3]{2x^2 - x^3} = \lim_{x\to+\infty} \sqrt[3]{x^3(\frac{2}{x} - 1)} = \lim_{x\to+\infty} x\sqrt[3]{\frac{2}{x} - 1} = (+\infty) \cdot (-1) = -\infty$$

pa zaključujemo da funkcija nema horizontalnu asimptotu ni na desnoj strani. Funkcija nema uopće horizontalne asimptote.

Kose asimptote: na lijevoj strani vrijedi

$$\lim_{x\to-\infty}\frac{f(x)}{x}=\lim_{x\to-\infty}\sqrt[3]{2x^2-x^3}\cdot\frac{1}{x}=\lim_{x\to-\infty}\sqrt[3]{\frac{2}{x}-1}=-1\equiv k$$

Dalje,

$$\lim_{x\to-\infty}f(x)-kx=\lim_{x\to-\infty}\sqrt[3]{2x^2-x^3}+x=(+\infty-\infty)$$

$$\lim_{x\to-\infty}x\left(\sqrt[3]{\frac{2}{x}-1}+1\right)=(-\infty)\cdot 0$$

$$\lim_{x\to-\infty}\frac{\sqrt[3]{\frac{2}{x}-1}+1}{\frac{1}{x}}=\lim_{x\to-\infty}\frac{\frac{1}{3}\left(\frac{2}{x}-1\right)^{-2/3}\left(-\frac{2}{x^2}\right)}{-\frac{1}{x^2}}=\lim_{x\to-\infty}\frac{2}{3}\left(\frac{2}{x}-1\right)^{-2/3}=\frac{2}{3}\equiv l$$

Dakle, pravac$y=-x+\frac{2}{3}$je kosa asimptota na lijevoj strani. Na desnoj strani

$$\lim_{x\to+\infty}\frac{f(x)}{x}=\lim_{x\to+\infty}\sqrt[3]{2x^2-x^3}\cdot\frac{1}{x}=\lim_{x\to+\infty}\sqrt[3]{\frac{2}{x}-1}=1\equiv k$$

Dalje, $\lim_{x\to+\infty}f(x)-kx=\lim_{x\to+\infty}\sqrt[3]{2x^2-x^3}-x=\lim_{x\to+\infty}x(\sqrt[3]{\frac{2}{x}-1}+1)=\lim_{x\to+\infty}\frac{\sqrt[3]{\frac{2}{x}-1}+1}{\frac{1}{x}}$

$$\lim_{x\to+\infty}\frac{\frac{1}{3}\left(\frac{2}{x}-1\right)^{-2/3}\left(-\frac{2}{x^2}\right)}{-\frac{1}{x^2}}=\lim_{x\to+\infty}\frac{2}{3}\left(\frac{2}{x}-1\right)^{-2/3}=\frac{2}{3}\equiv l$$

Dakle, pravac$y=x+\frac{2}{3}$je kosa asimptota na desnoj strani.

3.2.6 Traženje ekstrema funkcije

Izračunajmo prvu derivaciju:

$$f'(x)=\frac{1}{3}(2x^2-x^3)^{-2/3}(4x-3x^2)=\frac{1}{3}\cdot\frac{4x-3x^2}{\sqrt[3]{(2x^2-x^3)^2}}$$

Područje definicije derivacije je $\mathcal{D}_{f'}\in\mathbb{R}\backslash\{0,2\}$. Dakle, dvije kritične točke funkcije su $x_1=0$ i $x_2=2$. Za $x\in\mathcal{D}_{f'}$ možemo skratiti x u brojniku i nazivniku, odnosno vrijedi

$f'(x)=\frac{1}{3}\cdot\frac{4-3x}{\sqrt[3]{x(2-x)^2}}$. Vidimo da je stacionarna točka (treća kritična točka) jednaka $x_3=\frac{4}{3}$. Dakle, imamo tri točke koje zadovoljavaju nužan uvjet ekstema, odnosno u kojima funkcija može imati lokalne ekstreme. Dovoljne uvjete ekstrema provjerit ćemo pomoću prve derivacije, odnosno provjeriti ćemo mijenja li derivacija predznak u kritičnim točkama. Suočavamo se sa tri slučaja:

1) Za $x < 0$ je brojnik veći od nule, a nazivnik manji od nule stoga je $f'(x) < 0$. Drugim riječima, funkcija f je strogo padajuća na intervalu $(-\infty, 0)$.

2) Za $x \in (0, \frac{4}{3})$ su i brojnik i nazvnik veći od nule stoga je $f'(x) > 0$. Drugim riječima, funkcija f je strogo rastuća na intervalu $(0, \frac{4}{3})$.

3) Za $x > \frac{4}{3}$ je brojnik manji od nule, a nazivnik veći od nule stoga je $f'(x) < 0$. Drugim riječima, funkcija f je strogo padajuća na intervalu $(\frac{4}{3}, +\infty)$.

Proučavajući ovaj slučaj možemo zaključiti: iz 1) i 2) slijedi da funkcija ima lokalni minimum u kritičnoj točki $x_1 = 0$, vrijednost lokalnog minimuma je $f(0) = 0$, zatim iz 2) i 3) slijedi da funkcija ima lokalni maskimum u kritičnoj točki $x_3 = \frac{4}{3}$, vrijednost lokalnog maksimuma je $f\left(\frac{4}{3}\right) = 2\sqrt[3]{\frac{4}{3}}$ i konačno iz 3) slijedi da funkcija nema lokalni ekstrem u kritičnoj točki $x_2 = 2$, jer prva derivacija ne mijenja predznak u toj točki, odnosno funkcija je strogo padajuća s obje strane te točke. Funkcija nema globalni maksimum ni globalni minimum jer je kodomena jednaka $\mathbb{R}$.

3.2.7 Traženje intervala monotonosti funkcije

Monotonost smo već preispitali u prošlom podnaslovu stoga znamo kako je funkcija strogo padajuća u intervalima $(-\infty, 0)$ i $\left(\frac{4}{3}, +\infty\right)$ i strogo rastuća u intervalu $\left(0, \frac{4}{3}\right)$.

3.2.8 Traženje intervala zakrivljenosti (konkavnost)

Izračunajmo drugu derivaciju:

$$f''(x) = \frac{1}{3} \cdot \frac{-3\sqrt[3]{x(2-x)^2} - \frac{4-3x}{3[x(2-x)^2]^{2/3}} + x \cdot 2(2-x)(-1)}{\sqrt[3]{x^2(2-x)^4}}$$

$$= \frac{1}{3} \cdot \frac{-3(x(2-x)^2) - \frac{1}{3}(4-3x)(2-x)(2-3x)}{\sqrt[3]{x^4(2-x)^8}}$$

$$= \frac{-8}{9\sqrt[3]{x^4(2-x)^5}}$$

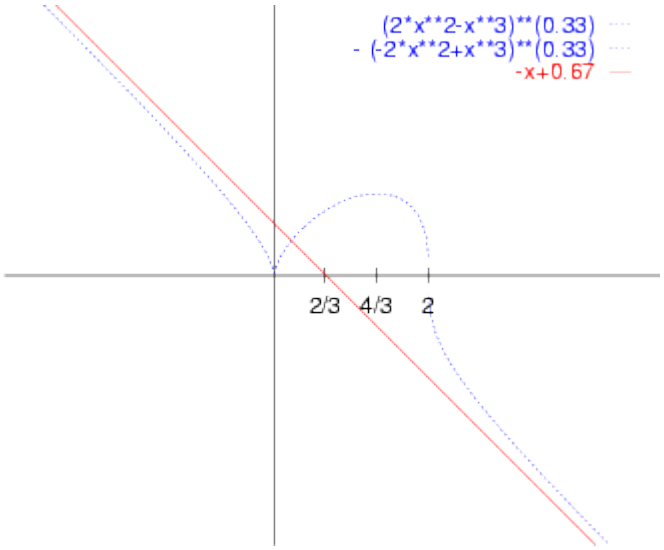
Uočavamo kako je predznak od f'' obrnut od predznaka izraza $2 - x$. Znači, za $x < 2$ je $f''(x) < 0$ pa je funkcija f konkavna. Za $x > 2$ je $f''(x) > 0$ pa je funkcija f konveksna.

3.2.9 Traženje točka infleksije

Prema podpoglavlju 2.2.2 zaključujemo da je točka infleksije funkcije $fx = 2$.

3.2.10Crtanje grafa funkcije

Koristeći sve dobivene rezultate računanja možemo konačno nacrtati graf funkcije. Graf funckije je nacrtan pomoću programa Gnuplot koji možete vidjeti na slici 3.1.



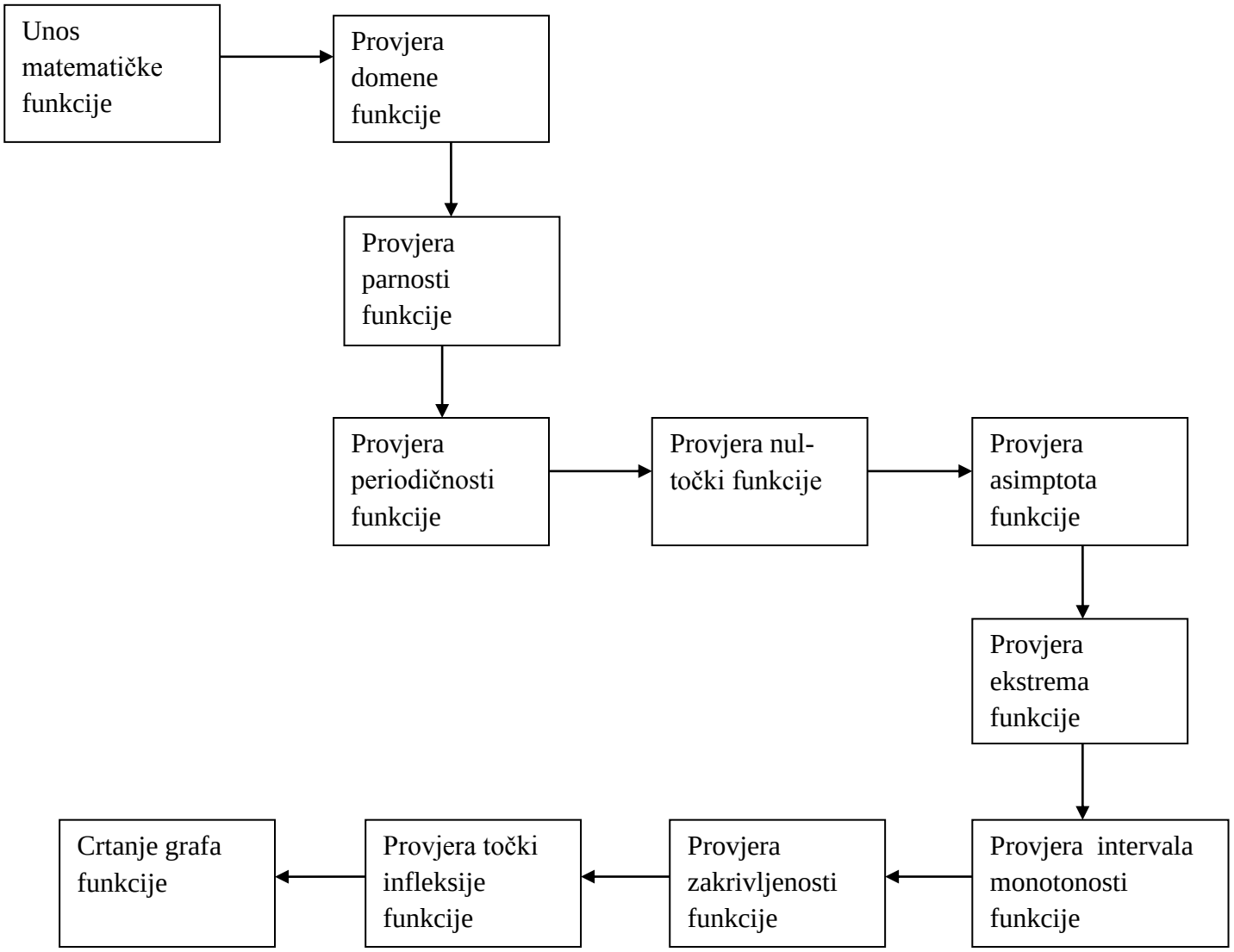
Sl.3.1: graf funkcije dobiven programom Gnuplot

4. PRIMJENA U PROGRAMSKOM JEZIKU C#

4.1 Zadatak programa

Zadatak programa je da korisnik unese proizvoljnu funkciju, te da izračuna cijeloviti tijek te funkcije.

4.2 Rad programa

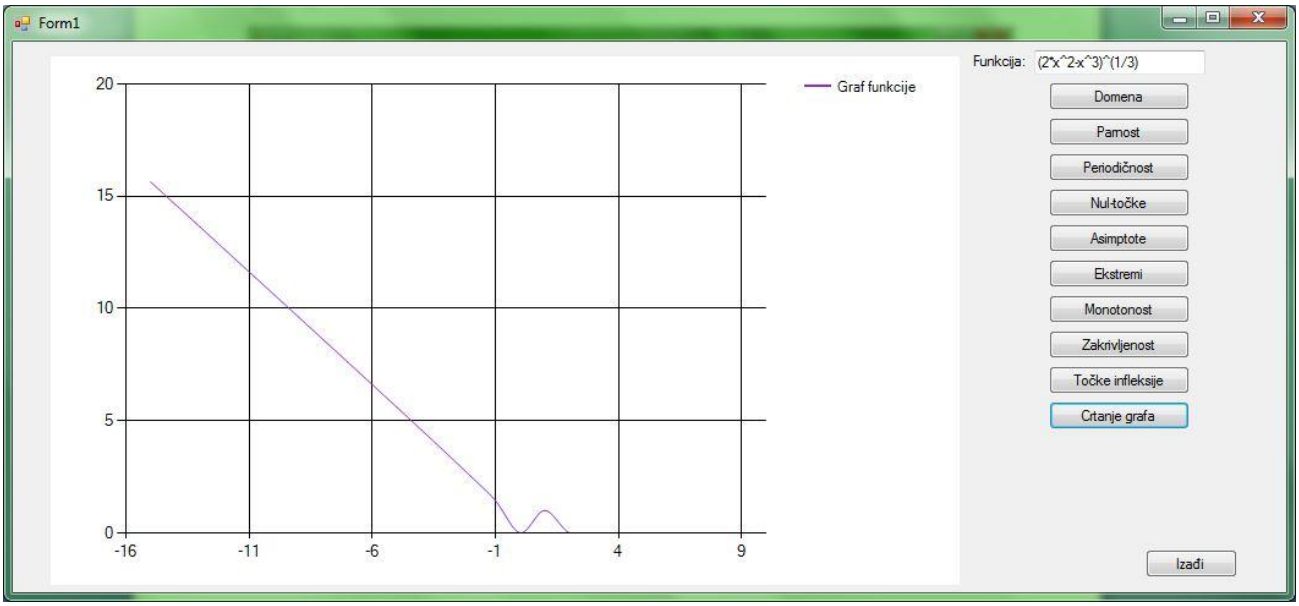


Sl 4.2: Dijagram toka programa

Program je napisan kao *Windows forms application*, odnosnoprogram koji koristi prozore s kojim korisnik međudjeluje. Pri pokretanju programa otvori se prozor kao što je prikazano na sljedećoj slici.

Sl.4.3: Početak rada programa

Od korisnika se traži unos matematičke funkcije u za to predviđen prostor. Nakon unosa funkcije korisnik može provjeriti tijek unesene funkcije tako što pritisne gumbove koji označavaju određeni dio tijeka funkcije. Na slijedećoj slici je prikazan unos funkcije koju smo uzeli kao primjer te nacrtan njen graf funkcije.



Sl.4.4: Graf unesene funkcije

Pri unosu funkcije se matematičke operacije poput korijenovanja i kvadriranja moraju unositi s znakom ^i različiti korijeni se moraju napisati u zagradama npr. treći korijen od *x* se piše kao $x\text{^}(1/3)$. Nakon pritiska jednog od gumbova iskoči prozor u kojemu je prikazan rezultat odabranog dijela tijeka funkcije. Na slijedećim slikama su prikazani rezultati tijeka funkcije.

Sl.4.5: Domena funkcije

Na Sl.4.5 vidimo skočni prozor koji nam govori koja je domena upisane funkcije. Skočni prozor je nastao pritiskom na gumb "Domena". Na sličan način će nam program preko skočnih prozora dati rješenja ostalih dijelova tijeka funkcije kao što su parnost, periodičnost, nultočke, ekstremi i slično.

4.3 Opis koda programa

Kod programa sastoji se od glavnog dijela programa, metode koja parsira *textbox* u koji se upisuje matematička funkcija, metode koja računa treći korijen, metode koja derivira i 11 funkcija koje se pozivaju kada korisnik klikne na gumbove. 10 gumbova se koriste za tijek funkcije, a zadnji gumb je gumb predviđen za izlaz iz programa.

Kod metode za određivanje domene funkcije:

```
privatevoid btdomena_Click(object sender, EventArgs e)
        {
string funk = funkbox.Text.ToString();
for(int i=0;i<funk.Length;i++)
                {
if(funk[i]=='/') //provjera nazivnika
                    {
if(funk[i+1]=='x') MessageBox.Show("Domena funkcije je R bez broja 0.");
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '0')
//pretražuje ako se nakon / nalazi (x+'0')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '0')
//pretražuje ako se nakon / nalazi (x-'0')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '1')
//pretražuje ako se nakon / nalazi (x+'1')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '1')
//pretražuje ako se nakon / nalazi (x-'1')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '2')
//pretražuje ako se nakon / nalazi (x+'2')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '2')
//pretražuje ako se nakon / nalazi (x-'2')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '3')
//pretražuje ako se nakon / nalazi (x+'3')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '3')
//pretražuje ako se nakon / nalazi (x-'3')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '4')
//pretražuje ako se nakon / nalazi (x+'4')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '4')
//pretražuje ako se nakon / nalazi (x-'4')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '5')
//pretražuje ako se nakon / nalazi (x+'5')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '5')
//pretražuje ako se nakon / nalazi (x-'5')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '6')
//pretražuje ako se nakon / nalazi (x+'6')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
```

```
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '6')
//pretražuje ako se nakon / nalazi (x-'6')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '7')
//pretražuje ako se nakon / nalazi (x+'7')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '7')
//pretražuje ako se nakon / nalazi (x-'7')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '8')
//pretražuje ako se nakon / nalazi (x+'8')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '8')
//pretražuje ako se nakon / nalazi (x-'8')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&& funk[i + 4] == '9')
//pretražuje ako se nakon / nalazi (x+'9')
MessageBox.Show("Domena funkcije je R kada x nije -" + funk[i + 4]);
elseif (funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&& funk[i + 4] == '9')
//pretražuje ako se nakon / nalazi (x-'9')
MessageBox.Show("Domena funkcije je R kada x nije " + funk[i + 4]);
                    }
elseif (funk[i] == '^') //provjera potencije ili korijena
                    {
if (funk[i + 1] == '('&& funk[i + 2] == '1'&& funk[i + 3] == '/'&& funk[i + 4] == '2')
                        {
if (funk[i - 1] == 'x')
MessageBox.Show("Domena je [0,inf>");
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '1')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '2')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '3')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '4')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '5')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '6')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '7')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '8')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '+'&& funk[i - 2] == '9')
MessageBox.Show("Domena funkcije je [-" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '1')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
```

```
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '2')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '3')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '4')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '5')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '6')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '7')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '8')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseif (funk[i - 5] == '('&& funk[i - 4] == 'x'&& funk[i - 3] == '-'&& funk[i - 2] == '9')
MessageBox.Show("Domena funkcije je [" + funk[i - 2] + ",inf>."); //inf stoji za
beskonačnost
elseMessageBox.Show("Domena funkcije je [0,inf>");

                        }
elseif (funk[i + 1] >= 1 || funk[i + 1] == '0')
MessageBox.Show("Domena funkcije je R.");
if (funk[i] == 'l'&& funk[i + 1] == 'o'&& funk[i + 2] == 'g') //provjera logaritma
MessageBox.Show("Domena funkcije je <0,inf>");
if (funk.Contains("arcsinx") || funk.Contains("arccosx")) //provjera arcsin i arccos
funkcije
MessageBox.Show("Domena funkcije je [-1,1].");
if (funk.Contains("sinx") || funk.Contains("cosx")) //provjera sinusne i kosinusne funkcije
MessageBox.Show("Domena funkcije je [-1,1].");
if (funk.Contains("tanx")) // provjera tangens funkcije
MessageBox.Show("Domena funkcije je R bez {pi/2+k*pi}");
if(funk.Contains("ctgx")) // provjera kotangens funkcije
MessageBox.Show("Domena funkcije je R bez {k*pi}");
                    }
elseif (funk[i] == 'x'&& (funk[i + 1]=='+' || funk[i + 1]=='-' || funk[i + 1]=='*') &&
(funk[i - 1] == '+' ||funk[i - 1] == '-' ||funk[i - 1] == '*' ))
MessageBox.Show("Domena funkcije je R.");
                }
        }
```

Gumb btdomena_Click računa domenu funkcije iako je ograničen za neke slučajeve. U kodu se koristi grananje kako bi se provjerilo nekoliko uvjeta za razne nazivnike i za druge matematičke funkcije kod kojih su izražene posebne domene.

Kod metode za određivanje parnosti funkcije:

```
privatevoid parnost_Click(object sender, EventArgs e)
        {
if (function((-1)) == function(1))
MessageBox.Show("Funkcija je parna.");
elseif (function(1) == (-1) * function(1))
MessageBox.Show("Funkcija je neparna.");
elseMessageBox.Show("Funkcija nije niti parna niti neparna.");
        }
```

Gumb parnost_Click vrlo jednostavno računa parnost funkcije. On koristi već napisanu metodu koja parsira te računa zadanu matematičku funkciju. U metodu unosi broj 1 kao vrijednost *x* te uspoređuje s istom funkcijom kada je vrijednost *x* jednaka *-1*.

Kod metode za određivanje periodičnosti funkcije:

```
privatevoid period_Click(object sender, EventArgs e)
        {
if (funkbox.Text.Contains("sin") || funkbox.Text.Contains("cos") ||
funkbox.Text.Contains("tan"))
MessageBox.Show("Funkcija je periodična");
elseMessageBox.Show("Funkcija nije periodična jer ne sadrži trig funkcije.");
        }
```

Gumb period_Click provjerava ako je korisnik uneo jednu od trigonometrijskih funkcija jer su trigonometrijske funkcije same po sebi periodične te stvaraju cijelu funkciju periodičnom.

Kod metode za određivanje nultočki funkcije:

```
privatevoid nult_Click(object sender, EventArgs e)
        {
string funk = funkbox.ToString();

double a=0;
for (int i = 0; i < funk.Length;i++ )
            {
if (funk[i] != 0)
                {
if (funk[i] == 'x'&& funk[i - 3] == 'l'&& funk[i - 2] == 'o'&& funk[i - 1] == 'g') //
provjera nultočke logaritamske funkcije
MessageBox.Show("Nultočka funkcije logx je 1.");
elseif (funk[i] == 'x'&& funk[i - 3] == 's'&& funk[i - 2] == 'i'&& funk[i - 1] == 'n') //
provjera nultočke sinus funkcije
MessageBox.Show("Nultočke funkcije sinx se kreću za svaki pi pa možemo reći da se nultočke
kreću po pravilu k*pi.");
elseif (funk[i] == 'x'&& funk[i - 3] == 'c'&& funk[i - 2] == 'o'&& funk[i - 1] == 's') //
provjera nultočke kosinus funkcije
MessageBox.Show("Nultočke funkcije cosx se kreću za svaki pi/2 pa možemo reći da se nultočke
kreću po pravilu k*pi/2.");
elseif (funk[i] == 'x'&& funk[i - 3] == 't'&& funk[i - 2] == 'a'&& funk[i - 1] == 'n') //
```

```
provjera nultočke tangens funkcije
MessageBox.Show("Nultočke funkcije tanx se kreću za svaki pi pa možemo reći da se nultočke
kreću po pravilu k*pi.");
elseif (funk[i] == 'x'&& funk[i - 3] == 'c'&& funk[i - 2] == 't'&& funk[i - 1] == 'g') //
provjera nultočke kotangens funkcije
MessageBox.Show("Nultočke funkcije sinx se kreću za svaki pi pa možemo reći da se nultočke
kreću po pravilu k*pi/2.");
                }
elseif(funk[i]=='x'&& funk[i+1]=='^'&& funk[i+2]=='2')
                {
if(funk[i+4]=='x'&& funk[i+5]=='^'&& funk[i+6]=='3'&& funk[i+3]=='+')
                    {
                        funk = "x^2*(1+x)";
MessageBox.Show("Nul točke funkcije su 0 i -1");
                    }
elseif(funk[i+4]=='x'&& funk[i+5]=='^'&& funk[i+6]=='3'&&funk[i+3]=='-')
                    {
                        funk = "x^2*(1-x)";
MessageBox.Show("Nul točke funkcije su 0 i 1");
                    }
                }
elseif (funk[i] == '2'&& funk[i + 1] == '*'&& funk[i+2] == 'x'&& funk[i + 3] == '^'&& funk[i
+ 4] == '2')
                {
if (funk[i + 6] == 'x'&& funk[i + 7] == '^'&& funk[i + 8] == '3'&& funk[i + 5] == '+')
                    {
                        funk = "x^2*(2+x)";
MessageBox.Show("Nul točke funkcije su 0 i -2");
                    }
elseif (funk[i + 6] == 'x'&& funk[i + 7] == '^'&& funk[i + 8] == '3'&& funk[i + 5] == '-')
                    {
                        funk = "x^2*(2-x)";
MessageBox.Show("Nul točke funkcije su 0 i 2");
                    }
                }
            }
if (funk[1] != '('&& funk[2] != '2'&& funk[3] != '*'&& funk[4] != 'x'&& funk[5] != '^'&&
funk[6] != '2'&& funk[7] != '-')
            {
for (int j = -100000; j < 100000; j++) //provjera ekstrema za nultočke
                {
                    a = j * 0.001;
if (function(a) == 0)
                    {
MessageBox.Show("Nultočka funkcije je " + a + " .");
                    }
                }
            }
        }
```

Gumb nult_Click prvo provjerava ako je korisnik unio neke od elementrarnih funkcija ili ako je unio neku od posebnih slučajeva. Ako nije onda se provjerava cijeli korisnikov unos te se traže točke gdje je funkcija jednaka nuli, tj. nultočke funkcije.

Kod metode za određivanje asimptota funkcije:

```
privatevoid asimpt_Click(object sender, EventArgs e)
        {
string funk = funkbox.ToString();
int d=0; //oznaka za domenu, jer se provjerava je li je domena cijeli skup R
double h1=0; double h2=0; //horizontalne asimptote
for(int i=0; i<funk.Length;i++) // provjera vertikalne asimptote
            {
if(funk[i]=='^')
                {
if (funk[i + 1] >= 1 || funk[i + 1] == '0')
                        d = 1;
elseif (funk[i + 1] == '('&& funk[i + 2] == '1'&& funk[i + 3] == '/'&& funk[i + 4] >= 51)
                        d = 1;
                }
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '1')
MessageBox.Show("Vertikalna asimptota je -1");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '2')
MessageBox.Show("Vertikalna asimptota je -2");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '3')
MessageBox.Show("Vertikalna asimptota je -3");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '4')
MessageBox.Show("Vertikalna asimptota je -4");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '5')
MessageBox.Show("Vertikalna asimptota je -5");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '6')
MessageBox.Show("Vertikalna asimptota je -6");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '7')
MessageBox.Show("Vertikalna asimptota je -7");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '8')
MessageBox.Show("Vertikalna asimptota je -8");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '+'&&
funk[i + 4] == '9')
MessageBox.Show("Vertikalna asimptota je -9");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '1')
MessageBox.Show("Vertikalna asimptota je 1");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '2')
MessageBox.Show("Vertikalna asimptota je 2");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '3')
MessageBox.Show("Vertikalna asimptota je 3");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '4')
MessageBox.Show("Vertikalna asimptota je 4");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
```

```
funk[i + 4] == '5')
MessageBox.Show("Vertikalna asimptota je 5");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '6')
MessageBox.Show("Vertikalna asimptota je 6");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '7')
MessageBox.Show("Vertikalna asimptota je 7");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '8')
MessageBox.Show("Vertikalna asimptota je 8");
elseif (funk[i] == '/'&& funk[i + 1] == '('&& funk[i + 2] == 'x'&& funk[i + 3] == '-'&&
funk[i + 4] == '9')
MessageBox.Show("Vertikalna asimptota je 9");
                } if(d==1)
                {
MessageBox.Show("Domena je R stoga funkcija nema vertikalnu asimptotu.");
                }
//provjera horizontalne asimptote
for(int i=0;i<100000;i++)
            {
                h1+=function(i);
if (h1 >Math.Pow(2, 31))
break;
            }
if(h1>100)
MessageBox.Show("Ne postoji lijeva horizontalna asimptota.");
elseif(h1>=0 && h1 <=100) MessageBox.Show("Lijeva horizontalna asimptota je " + h1);
elseMessageBox.Show("Ne postoji lijeva horizontalna asimptota.");
for(int i=0; i>-100000;i--)
            {
                h2-=function(i);
            }
if(h2<-100)
MessageBox.Show("Ne postoji desna horizontalna asimptota.");
elseMessageBox.Show("Desna horizontalna asimptota je " + h2);
//provjera kose asimptote
string funk2=funk + "/x";
double k1 = 0.00, b1 = 0.00, k2 = 0.00, b2 = 0.00;

            k1 = function(-1000) / (-1000); // lijeva asimptota, stavljen je broj -1000 kako
bi se približno odredio broj
            b1 = function(-1000) - (k1 * (-1000));
if (b1 == 0)
                b1 = (Derivacija(function(-1000)) - k1) * (-1000);
MessageBox.Show("Lijeva kosa asimptota je y=" + Math.Round(k1, 2) + "*x+" +
Math.Round(Math.Abs(b1), 2));

            k2 =(function(1000) / (1000)); // desna asimptota, stavljena je funkcija
Math.Abs kako bi dobili pozitivnu vrijednost jer desna asimptota ne može biti na lijevoj
strani koordinatnog sustava
            b2 = function(1000) - (k2 * (1000));
if(b2==0)
                b2 = (Derivacija(function(1000)) - k2) * 1000;
MessageBox.Show("Desna kosa asimptota je y=" + Math.Round(Math.Abs(k2), 2) + "*x+" +
Math.Round(Math.Abs(b2), 2));
        }
```

Gumb asim_Click provjerava ako unesena funkcija sadrži vertikalnu, horizontalne ili kose asimptote. Za vertikalne asimptote postoje slučajevi kada postoje vertikalne asimptote i kada one ne postoje i zbog toga se provjerava domena funkcije opet kako bi se vertikalna asimptota mogla odrediti. Horizontalna asimptota se provjerava kroz petlju koja računa sto tisuća brojeva kroz unesenu funkciju te time traži broj koji bi predstavljao horizontalnu asimptotu. Budući da postoji lijeva i desna horizontalna asimptota onda program provjerava pozitivnih sto tisuća brojeva i negativnih sto tisuća brojeva. U slučaju da je broj prevelik ili ga nema u traženom rasponu, ako je broj negativan, onda se smatra da ne postoji desna ili lijeva asimptota. Kosa asimptota se provjerava tako što se uzima broj s kojim se može približiti rješenjeizraza, u ovom slučaju je odabran broj tisuću. Ako se ne može naći kosa asimptota onda se derivira izraz kako bi uspjeli doći do rješenja.

Kod metode za određivanje ekstrema funkcije:

```
privatevoid ekstrm_Click(object sender, EventArgs e)
        {
string funk = funkbox.ToString();
double nul = 0; //nultočka
double a = 0; //varijabla pomoću koje provjeravamo decimale
if (funk.Contains("(2*x^2-x^3)^(1/3)")) //provjera ekstrema za primjer
                {
if (Derivacija(0) > 0)
MessageBox.Show("U nultočki 0 se nalazi lokalni maksimum.");
elseMessageBox.Show("U nultočki 0 se nalazi lokalni minimum.");
if (Derivacija(4 / 3) < 0)
MessageBox.Show("U stacionarnoj točki 4/3 se nalazi lokalni minimum.");
elseif(Derivacija(4 / 3) > 0) MessageBox.Show("U stacionarnoj točki 4/3 se nalazi lokalni
maksimum.");
if (Derivacija(2) < 0) MessageBox.Show("U nultočki 2 nema lokalnih ekstrema jer joj
derivacija ne mijenja predznak.");
elseif (Derivacija(2) > 0) MessageBox.Show("U nultočki 2 se nalazi lokalni maksimum.");
elseif (Derivacija(2) == 0) MessageBox.Show("U nultočki 2 se nalazi lokalni minimum");
                }
elseif (funk[1] != '('&& funk[2] != '2'&& funk[3] != '*'&& funk[4] != 'x'&& funk[5] != '^'&&
funk[6] != '2'&& funk[7] != '-')
            {
for (int j = -100000; j < 100000; j++) //provjera ekstrema za nultočke
                {
                    a = j * 0.001;
if (function(a) == 0)
                    {
                        nul = a;
if (Derivacija(nul) < 0)
MessageBox.Show("U nultočki " + nul + " se nalazi lokalni minimum.");
elseif (Derivacija(nul) > 0) MessageBox.Show("U nultočki " + nul + " se nalazi lokalni
maksimum.");
                    }
                }
            }
        }
```

Gumb ekstrm_Click se sastoji od dva dijela. Prvi dio metode provjerava naš primjer jer je on specifičnog unosa, a i on sadrži stacionarnu točku. Ovdje deriviramo funkciju te provjeravamo ako je derivacija funkcije u nultočkama veća ili manja od nule i na temelju toga zaključujemo je li je nultočka funkcije lokalni minimum ili lokalni maksimum funkcije. Može se vidjeti kako druga nultočka ima specifičan slučaj te iako što ima derivaciju manju od nule se ne gleda kao lokalni minimum funkcije. Također se može primjetiti kako je ovdje iskorištena starcionarna točka koja se dobije derviranjem funkcije te je ona lokalni maksimum funkcije. U drugom dijelu metode gumba se provjerava postoje li nultočke te ovisno o iznosu derivacije se zaključuje o tome postoje li lokalni minimumi ili lokalni maksimumi.

Kod metode za određivanje monotonosti funkcije:

```
privatevoid monoton_Click(object sender, EventArgs e)
        {
string funk = funkbox.ToString();
int i;
double nul = 0, a;
if (funk.Contains("(2*x^2-x^3)^(1/3)")) //provjera monotonosti funkcije za primjer
            {
if (Derivacija(0) < 0)
MessageBox.Show("Funkcija do točke 0 strogo raste.");
elseMessageBox.Show("Funkcije do točke 0 strogo pada.");
if (Derivacija(4 / 3) < 0)
MessageBox.Show("Funkcija do točke 4/3 strogo pada.");
elseMessageBox.Show("Funkcija do točke 4/3 strogo raste.");
if (Derivacija(2) < 0) MessageBox.Show("Funkcija do točke 2 strogo pada.");
elseMessageBox.Show("Funkcija do točke 2 strogo raste.");
            }
else
for (i = -100000; i < 100000; i++) //traženje nultčke i provjeravanje rasta i pada funkcije
            {
                a = i * 0.001;
if (function(a) == 0)
                {
                    nul = a;
if (Derivacija(nul) < 0)
MessageBox.Show("Funkcija do točke " + nul + " strogo pada.");
elseif (Derivacija(nul) > 0) MessageBox.Show("Funkcija do točke " + nul + " strogo raste.");
                }
            }
        }
```

Gumb monoton_Click se također sastoji iz dva dijela. Prvi dio ispituje monotonost za funkciju iz primjera jer ona sadrži stacionarnu točku. Oba dijela funkcije ispituju je li je derivacija funkcije u nultočki veća ili manja od nule te se na temelju toga se određuje strogi rast i strogi pad funkcije nakon tih točki.

Kod metode za određivanje zakrivljenosti funkcije:

```
privatevoid zak_Click(object sender, EventArgs e)
        {
string funk = funkbox.ToString();
double a;
int i = 0;
if (funk.Contains("logx")) //provjera zakrivljenosti logaritamske funkcije
MessageBox.Show("Funkcija je konkavna.");
elseif (funk.Contains("sinx")) //provjera zakrivljenosti sinusoidne funkcije
MessageBox.Show("Funkcija se kreće po pravilu k*pi, stoga je u jednom periodu i konkavna i
konveksna.");
elseif (funk.Contains("cosx")) //provjera zakrivljenosti kosinusoidne funkcije
MessageBox.Show("Funkcija se kreće po pravilu k*pi/2, stoga je u jednom periodu i konkavna i
konveksna.");

elsefor (i = -100000; i < 100000; i++) //provjeravanje zakrivljenosti funkcije
                {
                    a = i * 0.001;
if (function(a) == 0)
                    {
if ((Derivacija(a)) < 0)
MessageBox.Show("Funkcija je za x manji od " + a + " konkavna.");
elseif ((Derivacija(a)) > 0)
MessageBox.Show("Funkcija je za x veći od " + a + " konveksna.");
                    }
                }
        }
```

Gumb zak_Click provjerava zakrivljenost funkcije, tj. ako je funkcija na dijelovima ili u cijelosti konveksna ili konkavna. Početak koda provjerava ako unesena funkcija sadrži neke od elementarnih funkcija koje utječu na izgled funkcije. Ako ne sadrže onda se traže nultočke funkcije i provjerava se uvjetom konveksnost ili konkavnost funkcije pomoću derivacije.

Kod metode za određivanje točki infleksije:

```
privatevoid infleks_Click(object sender, EventArgs e)
        {
double a;
int i = 0;
for (i = -100000; i < 100000; i++) //traženje nultčke i provjeravanje rasta i pada funkcije
            {
                a = i * 0.001;
if (function(a) == 0)
                {
if ((Derivacija(a)) < 0 && Derivacija(a+0.1)<0)
MessageBox.Show("Točka infleksije funkcije je u " + a + " .");
elseif ((Derivacija(a)) > 0 && Derivacija(a+0.1)>0)
MessageBox.Show("Točka inflekcije funkcije je u " + a + " .");
                }
            }
        }
```

Gumb infleks_Click provjerava ako postoje točke infleksije funkcije. Budući da su točke infleksije točke u kojima se mijenja zakrivljenost funkcije onda smo ih provjeravali pomoću deriviranja funkcije i traženjem nultočki funkcije.

Kod metode za crtanje funkcije:

```
privatevoid crt_Click(object sender, EventArgs e)
        {
string funk = funkbox.Text.ToString();
for (int i = -10; i < 10; i++)
            {
                chart1.Series[0].Points.AddXY(i, function(i)); //crtanje funkcije
            }
        }
```

Gumb crt_Click crta unesenu funkciju. Crtanje je izvedeno pomoću *chart* alata u *Visual Studio 2013*. Pokrećemo petlju koja koristi metodu dodavanja *x* i *y* koordinata na naš *chart* te kako bi dodali te koordinate koristimo našu funkciju koja parsira i vraća rješenje funkcije na uneseni broj, tj. za uneseni *x* vraća *y*.

Kod metode za parsiranje unesene funkcije:

```
double function(double x) //metoda matematičke funkcije
        {
string funk = funkbox.Text.ToString();
int b=1;
int koef = 0;
int koef_uk = 0;
double y=0;

for (int i = 0; i < funk.Length; i++)
            {
if (funk[i] == '('&& funk[i + 1] == '2'&& funk[i + 2] == '*'&& funk[i + 3] == 'x'&& funk[i +
4] == '^'&& funk[i + 5] == '2'&& funk[i + 6] == '-'&& funk[i + 7] == 'x'&& funk[i + 8] ==
'^'&& funk[i + 9] == '3'&& funk[i + 10] == ')'&& funk[i + 11] == '^'&& funk[i + 12] == '('&&
funk[i + 13] == '1'&& funk[i + 14] == '/'&& funk[i + 15] == '3'&& funk[i + 16] == ')') //
uvjet za primjer
return y = CubeRoot(2 * Math.Pow(x, 2) - Math.Pow(x, 3));
elseif (funk[i] == 'x')
                {
int j = i;
if (i == 0) b = 0; // "zastavica" se postavlja u nulu jer nepoznanica nema predznak
else
                    {
while (funk[j - 2] == '0' || funk[j - 2] == '1' || funk[j - 2] == '2' || funk[j - 2] == '3'
|| funk[j - 2] == '4' || funk[j - 2] == '5' || funk[j - 2] == '6' || funk[j - 2] == '7' ||
funk[j - 2] == '8' || funk[j - 2] == '9'&& funk[j - 1] == '*')
                        {
if (i > 2 && funk[j - 3] >= 48 && funk[j - 3] <= 57) // provjeravanje druge znamenke
koeficijenta
```

```
                            {
if (i > 3 && funk[j - 4] >= 48 && funk[j - 4] <= 57) // provjeravanje treće znamenke
koeficijenta
                                {
Int32.TryParse(funk[j - 4].ToString(), out koef);
                                    koef_uk = koef * 100; // stavljanje prve znamenke u
funkciju
                                }
Int32.TryParse(funk[j - 3].ToString(), out koef);
                                koef_uk += koef * 10; // stavljanje druge znamenke u
funkciju
                            }
Int32.TryParse(funk[j - 2].ToString(), out koef);
                            koef_uk += koef; // stavljanje treće znamenke u funkciju
                            j++;
                        }
                    }
if (i != funk.Length)
                    {
if (funk[i + 1] == '^')
                        {
if (funk[i + 2] == '2')
                                y += koef_uk * Math.Pow(x, 2);
elseif (funk[i + 2] == '3')
                                y += koef_uk * Math.Pow(x, 3);
elseif (funk[i + 2] == '4')
                                y += koef_uk * Math.Pow(x, 4);
elseif (funk[i + 2] == '5')
                                y += koef_uk * Math.Pow(x, 5);
elseif (funk[i + 2] == '('&& funk[i + 4] == '/'&& funk[i + 6] == '6')
                            {
if (funk[i + 3] == '1'&& funk[i + 5] == '2')
                                    y += koef_uk * Math.Sqrt(x);
elseif (funk[i + 3] == '1'&& funk[i + 5] == '3')
                                    y += koef_uk * Math.Pow(x, 0.3333333333);
                            }
                        }
elseif (funk[i - 3] == 's'&& funk[i - 2] == 'i'&& funk[i - 1] == 'n')
                            y += Math.Sin(x);
elseif (funk[i - 3] == 'c'&& funk[i - 2] == 'o'&& funk[i - 1] == 's')
                            y += Math.Cos(x);
elseif (funk[i - 3] == 'l'&& funk[i - 2] == 'o'&& funk[i - 1] == 'g')
                            y += Math.Log10(x);
elseif (funk[i - 2] == 'l'&& funk[i - 1] == 'n')
                            y += Math.Log(x);
elseif (funk[i - 2] == 't'&& funk[i - 1] == 'g')
                            y += Math.Tan(x);
                    }

                }
            }
return y; //unos funkcije
        }
```

Metoda function čita funkciju koju smo unijeli u *textbox* te prvo provjerava ako smo unijeli funkciju koja je korištena kao primjer za ovaj rad iz razloga što je nemoguće parsirati tu funkciju. Inače

metoda provjerava ako postoji nepoznanica, tj. *x* te zatim provjerava koeficijent ispred *x* do troznamenkastog broja, zatim provjerava ako ta nepoznanica ima neki eksponent i ako postoje neke od elementarnih funkcija kao npr. sinus, kosinus, logaritam, prirodni logaritam i tangens. Nakon pretraživanja i parsiranja se vraća *y* kao rezultat unesenog broja i parsirane funkcije.

Kod metode za određivanje derivacije funkcije:

```
double Derivacija(double x)
        {
string funk = funkbox.Text.ToString(); //stvaranje stringa koji sadrži matematičku funkciju
string koef_str = ""; // string koji zapisuje koeficijent x-a
string y_str = ""; // string koji zapisuje deriviranu funkciju

double y = 0;
int koef = 1;


int b = 1; // "zastavica" koja označava je li ima koeficijenta ili ne
for (int i = 0; i < funk.Length; i++) // pretraživanje stringa, tj. pretraživanje
matematičke funkcije
            {
if (funk[i] == 'x') // tražimo nepoznanicu
                {
int j = i;
if (i == 0) b = 0; // "zastavica" se postavlja u nulu jer nepoznanica nema predznak
else
                    {
while (funk[j - 2] == '0' || funk[j - 2] == '1' || funk[j - 2] == '2' || funk[j - 2] == '3'
|| funk[j - 2] == '4' || funk[j - 2] == '5' || funk[j - 2] == '6' || funk[j - 2] == '7' ||
funk[j - 2] == '8' || funk[j - 2] == '9'&& funk[j - 1] == '*')
                        {
if (i > 2 && funk[j - 3] >= 48 && funk[j - 3] <= 57) // provjeravanje druge znamenke
koeficijenta
                            {
if (i > 3 && funk[j - 4] >= 48 && funk[j - 4] <= 57) // provjeravanje treće znamenke
koeficijenta
                                {
                                    koef_str += funk[j - 4]; // stavljanje prve znamenke u
string
                                }
                                koef_str += funk[j - 3]; // stavljanje druge znamenke u
string
                            }
                            koef_str += funk[j - 2]; // stavljanje treće znamenke u string
                            j++;
                        }
Int32.TryParse(koef_str, out koef);
                        koef_str = "";
//MessageBox.Show(koef.ToString());
                    }
if (funk[i + 1] == '^') // tražimo potenciju
                    {
if (funk[i + 2] == '1') // provjeravamo ako je 'x na prvu'
                        {
```

```
                            y += koef * 1;
                            y_str += koef_str;
                        }
elseif (funk[i + 2] == '2') // provjeravamo ako je 'x na kvadrat'
                        {
                            y += koef * 2 * x;
                            y_str += "*2*x";
                        }
elseif (funk[i + 2] == '3') // provjeravamo ako je 'x na treću'
                        {
                            y += koef * 3 * Math.Pow(x, 2);
                            y_str += koef_str + "*3*x^2";
                        }
elseif (funk[i + 2] == '4') // provjeravamo ako je 'x na četvrtu'
                        {
                            y += koef * 4 * Math.Pow(x, 3);
                            y_str += koef_str + "*4*x^3";
                        }
elseif (funk[i + 2] == '5') // provjeravamo ako je 'x na petu'
                        {
                            y += koef * 5 * Math.Pow(x, 4);
                            y_str += koef_str + "*5*x^4";
                        }
elseif (funk[i + 2] == '('&& funk[i + 4] == '/'&& funk[i + 6] == ')') // slučaj ako postoji
korijen
                        {
if (funk[i + 3] == '1')
                            {
if (funk[i + 5] == '2')
                                {
                                    y += (1 / 2) * koef * Math.Pow(x, -0.5); //drugi korijen
                                    y_str += "(1/2)" + koef_str + "*x^(-1/2)";
                                }
elseif (funk[i + 5] == '3')
                                {
                                    y += (1 / 3) * koef * Math.Pow(x, -0.6666666); //treći
korijen
                                    y_str += "(1/3)" + koef_str + "*x^(-2/3)";
                                }
elseif (funk[i + 5] == '4')
                                {
                                    y += (1 / 4) * koef * Math.Pow(x, -0.75); //četvrti
korijen
                                    y_str += "(1/4)" + koef_str + "*x^(-3/4)";
                                }
                            }
                        }
                    }
if (funk[i - 3] == 'l'&& funk[i - 2] == 'o'&& funk[i - 1] == 'g')
                        y += (1 / (x * Math.Log(10)));
if (funk[i - 3] == 's'&& funk[i - 2] == 'i'&& funk[i - 1] == 'n')
                        y += Math.Cos(x);
if (funk[i - 3] == 'c'&& funk[i - 2] == 'o'&& funk[i - 1] == 's')
                        y += (-1) * Math.Sin(x);
if (funk[i - 2] == 't'&& funk[i - 1] == 'g')
                        y += 1 / (Math.Pow(Math.Cos(x), 2));
                }
            }
```

```
for(int i=0; i<1; i++) //složena derivacija primjera
            {
       if(funk[i]=='('&& funk[i+10]==')'&& funk[i+11]=='^'&& funk[i+12]=='('&&
funk[i+13]=='1'&& funk[i+14]=='/'&& funk[i+16]==')')
                     {
                      y=(4*x-3*Math.Pow(x,2))/(3*Math.Pow(Math.Pow(2*Math.Pow(x,2)-
Math.Pow(x,3),2), 0.3333333));
                    }

            }
return y;
        }
```

Metoda Derivacija radi slično pretraživanje *textbox* polja te za napisane slučajeve ako što su potencije, elementarne funkcije i sl. računa derivaciju i zbraja s prethodnim. Metoda je ograničena na određen broj potencija i nisu napisane sve potencije jer u prethodnoj funkciji nije omogućen sav mogući unos raznih funkcija. Za naš primjer, budući da je on složena derivacija, je napravljen kod koji točno izračunava derivaciju. Metoda prima broj koji se gleda kao *x* funkcije.

Kod metode za računanje trećeg korijena:

```
double CubeRoot(double x)
        {
if (x < 0)
return (-1)*Math.Pow(Math.Abs(x), (1.0 / 3.0));
elsereturnMath.Pow(x, (1.0 / 3.0));
        }
```

Metoda CubeRoot prima neki broj te provjerava ako je on manji od nule jer treći korijen može računati negativne brojeve, te ako je negativan onda rješenje množi s minus jedan, a ako nije onda normalno računa treći korijen. Ova metoda je napravljena zbog ograničenja *Visual Studio 2013* koji ne može računati treći korijen iz negativnih brojeva što je bilo potrebno za naš primjer.

5. ZAKLJUČAK

Tijek funkcije je postupak traženja područja definicije, parnosti, periodičnosti, nultočki, asimptota, ekstrema, monotonosti, zakrivljenosti i točki infleksije neke matematičke funkcije. Pomoću tijeka funkcije više saznajemo o samoj funkciji bez da ju nacrtamo.

Zadatak ovog završnog rada je bio napisati aplikaciju koja može točno odrediti tijek funkcije te nacrtati unesenu funkciju. Aplikacija je uspješno napravljena, no sadrži određena ograničenja zbog moguće kompleksnosti zadatka. Ograničenja koja su postavljena su vezana uz derivacije složenijih funkcija i složenijih metoda deriviranja kao što je deriviranje funkcije s nepoznanicama u brojniku i nazivniku. Zatim prilikom parsiranja funkcije nije bilo moguće parsirati funkciju koja ima izraz korijena preko dvije nepoznanice, prilikom traženja domene kod funkcije koja ima nepoznanicu u nazivniku je ograničeno do izraza $x+9$ i $x-9$. Kako bi se aplikacija riješila navedenih ograničenja potrebno je provesti više vremena na izradi ovakvog programa. Zadatak sam po sebi nije težak, no zahtjeva određenu količinu vremena kako bi se već prethodno znanje iz matematike moglo primjeniti u obliku programskog koda.

LITERATURA

[1] B. P. Demidovič i suradnici, ZADACI I RIJEŠENI PRIMJERI IZ MATEMATIČKE ANALIZE ZA TEHNIČKE FAKULTETE, Golden Marketing Tehnička Knjiga, Zagreb, 2003.

[2] Matematika 1, http://lavica.fesb.hr/mat1/predavanja/node122.html, pristupio 3.6.2017

[3] M. Pačar i M. Katalinić, Funkcije, kompleksni brojevi, polinomi, Dvostruka duga, 2012.

[4] M. Pačar i M. Katalinić, Diferencijalne jednadžbe s primjerom, Dvostruka duga, 2012.

[5] John Sharp, Microsoft Visual C# 2013 Step by Step, Microsoft Press, 2013.

[6] Steve McConnell, Code Complete Second Edition, Microsoft Press, 2004.

SAŽETAK

Cilj ovog rada je bio opisati funkcije, dijelove funkcije, tijek funkcije te njegove dijelove, prikaz izračunavanja tijeka funkcije i napisati program koji može uspješno riješiti tijek funkcije. Uzet je jedan riješen primjer tijeka funkcije te je napravljena aplikacija koja rješava taj primjer uz ostale moguće primjere. Prvo se izračunava područje definicije funkcije, zatim parnost ili neparnost funkcije, provjerava se je li je funkcija periodična, tj. ponavlja li se funkcija nakon određenog perioda. Zatim se provjeravaju nultočke funkcije, tj. točke u kojima funkcija siječe *x*-os, provjeravaju se asimptote koje su pravci koji vertikalno, horizontalno ili koso omeđuju i ograničuju funkciju. Nakon asimptota se provjerava ako postoje ekstremi, tj. najniže ili najviše točke nakon kojih funkcija raste ili pada, zatim se traži monotonost funkcije, zakrivljenost funkcije, točke infleksije i konačno se crta graf funkcije.

Ključne riječi: tijek funkcije, ekstremi, programski jezik C#, točke infleksije, nultočke

ABSTRACT

C# APLICATION FOR DRAWING A MATHEMATICAL FUNCTION AND DETERMINIG THE FLOW OF A FUNCTION

The aim of this paper was to describe functions, parts of functions, a flow of functions along with its parts, show the steps to calculate the flow of a function and write an application which can successfully determine the flow of a function. A solved example of the flow of the function was taken and an application which can calculate that example along with other examples was made. Firstly, it calculates the domain of a function, then parity or oddity of a function; it checks if a function is periodical, i.e. if it repeats itself after a specific period. Then, it calculates zero points of a function, i.e. points where the function cuts the x-axis; it checks asymptotes - the lines which can vertically, horizontally or diagonally bound and restrict a function. After asymptotes, it checks whether it has extremes, i.e. the lowest or the highest points after which a function rises or falls. Further, it checks monotony of a function, curvature of a function and inflection points. Finally, it draws a chart of the function.

Keywords: flow of a function, extreams, C# programming language, inflection points, zero points

ŽIVOTOPIS

Dominik Birtić rođen je u Osijeku, Republika Hrvatska, 6. siječnja 1996. godine. Pohađao je osnovnu školu "Retfala" u Osijeku i svih je osam razreda prošao s odličnim uspjehom.

Nakon osnovne škole, 2010. upisuje prirodoslovno-matematičku gimnaziju u Osijeku i završava sve razrede s vrlo dobrim uspjehom. 2014. godine je maturirao s dobrim uspjehom.

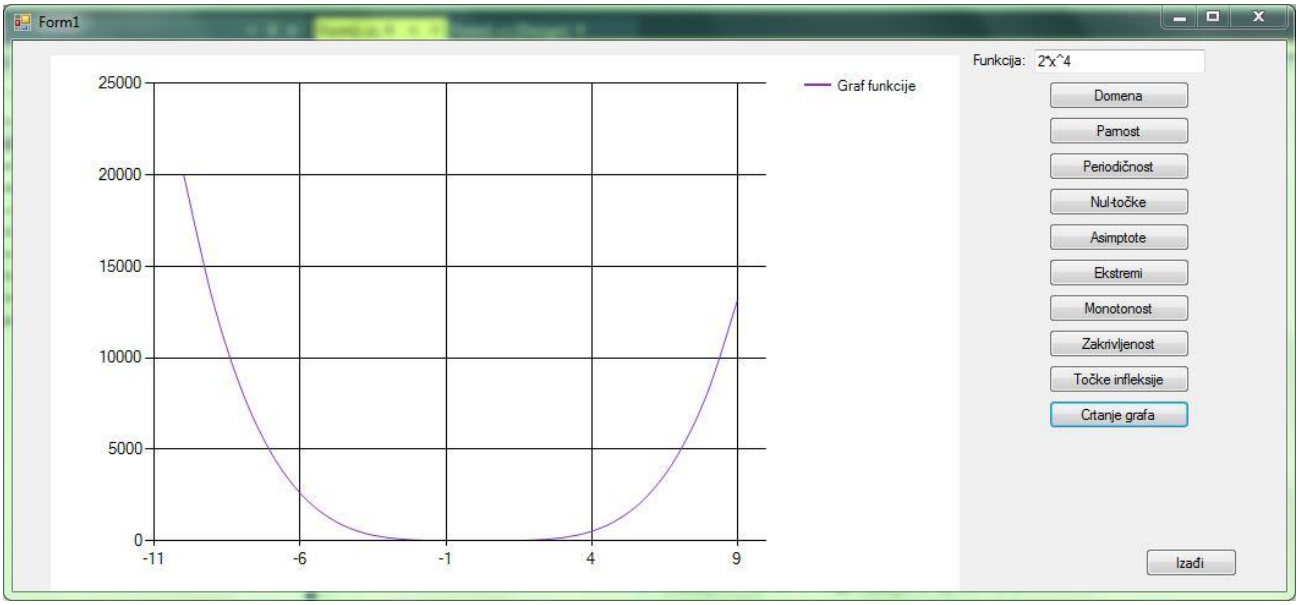
2014. godine upisuje Elektrotehnički fakultet u Osijeku, smjer računarstvo. Nakon završenog preddiplomskog studija planira upisati diplomski studij na istom fakultetu.

Vlastoručni potpis:

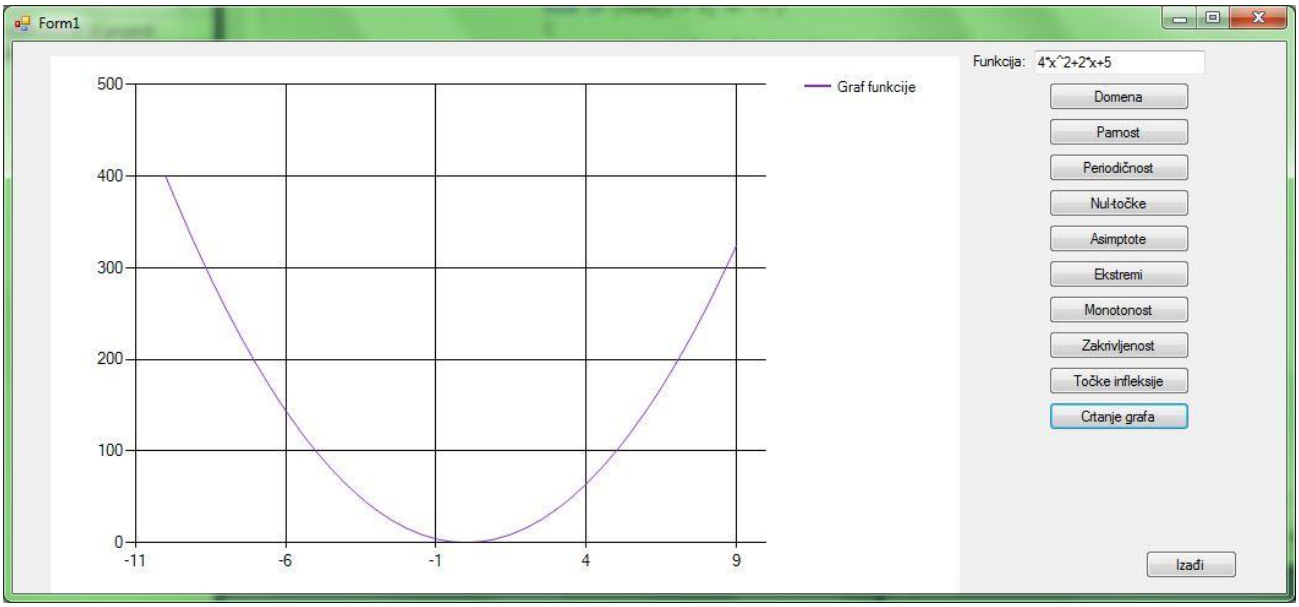
_____

Dominik Birtić

PRILOZI



P.4.1. Graf funkcije četvrtog reda



P.4.2. Graf funkcije drugog reda