KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Skaitiniai metodai ir algoritmai (P170B115)

Laboratorinių darbų ataskaita

Atliko:

IFF-1/4 gr. studentas

Mildaras Karvelis

2023 m. spalio 26 d.

Priėmė:

Prof. Barauskas Rimantas

KAUNAS 2023

#### TURINYS

1. Netiesinių lygčių sprendimas 3

1.1. f(x) daugianario sprendimas 3

1.2. g(x) daugianario grafiškas vaizdavimas 5

1.3. Programos kodo fragmentai 5

2. Šaknų atskyrimas skenavimo metodu. 7

2.1. f(x) ir g(x) grafikai ir intervalai 7

2.2. Programos kodo fragmentai 8

3. Šaknų tikslinimas skenavimo, ~~pusiaukirtos~~, Kvazi-Niutono (kirstinių) metodais. 9

3.1. f(x) daugianaris 9

3.2. g(x) funkcija 14

3.3. Programos kodo fragmentai 15

4. 2 dalis 20

4.1. g(x) ir TE tarpiniai grafikai 20

4.2. g(x) ir TE grafikai su reikalaujančiu tikslumu 21

4.3. TE analitinė išraiška 21

4.4. Sprendinai su skirtingu TE narių skaičiumi 22

4.5. Programos kodo fragmentai 24

# A white rectangular object with black text Description automatically generated with medium confidenceNetiesinių lygčių sprendimas

Sprendimo metodai: Pusiaukirtos ir Kvazi-Niutono (kirstinių)

## A graph on a grid Description automatically generatedf(x) daugianario sprendimas

1 pav. Daugianario tikslus įvertis

A graph with a line

Description automatically generated

pav. Daugianario grubaus įverčio grafikas

|  |  |
| --- | --- |
| Grubus lygties f(x) = 0 šaknų intervalo įvertis | [-6.999999999999999 ; 6.999999999999999] |
| Tikslesnis lygties f(x) = 0 šaknų intervalo įvertis | [-3.449489742783178 ; 4.746268656716417] |

lentelė Tikslūs ir grubūs įverčiai

## g(x) daugianario grafiškas vaizdavimas

A graph with a line and a red dot

Description automatically generated

pav. g(x) daugianario grafinis vaizdas

## Programos kodo fragmentai

|  |
| --- |
| function iverciai\_ir\_grafikai  clc, close all, clear all;  format long;  % ------------------------------------  % Daugianaris f(x)  f = @(x)(-0.67\*x.^4) + 2.51\*x.^3+2.27\*x.^2-4.02\*x-2.48;  %f = @(x)(0.67\*x.^4) - 2.51\*x.^3-2.27\*x.^2+4.02\*x+2.48;  f\_name = '-0.67x^4+2.51x^3+2.27x^2-4.02x-2.48';  % ------------------------------------  % Funkcija g(x)  g = @(x)exp(-1\*x.^2).\*sin(x.^2).\*(x+2);  g\_name = 'e^-^x\*sin(x^2)\*(x+2)';  % ------------------------------------  a = [0.67 -2.51 -2.27 4.02 2.48];  n = numel(a);  [R\_grub, R\_neig, R\_teig]=Reziai(n, a);  Grubus = [-R\_grub R\_grub]  Tikslesnis = [R\_neig R\_teig]  % ------------------------------------  % grafikų braižymas  grubus\_intervalas = -R\_grub:0.1:R\_grub;  tikslus\_intervalas = R\_neig:0.1:R\_teig;  % ------------------------------------  % f(x) grubus  figure(1); hold on; grid on;  % plot(-min(R\_grub, R\_neig), 0, 'bp', 'LineWidth', 2);  % plot(min(R\_grub,R\_teig),0,'bp', 'LineWidth', 2);  plot([-R\_grub,R\_grub],[0 0],'r\*', 'LineWidth', 2);  plot([R\_neig R\_teig], [0 0], 'bp', 'LineWidth', 2);  plot(grubus\_intervalas, f(grubus\_intervalas), 'k-', 'LineWidth', 2);  title(['f(x)=', f\_name, ' Grubus intervalas.']);  legend('Grubus šaknų intervalo įvertis', 'Tikslesnis šaknų intervalo įvertis', 'Daugianaris f(x)');  axis([-R\_grub R\_grub -R\_grub R\_grub]);  plot([-R\_grub, R\_grub], [0, 0], 'b'); % X ašies linija  % ------------------------------------  % f(x) tikslus  figure(2); hold on; grid on;  plot([-R\_grub,R\_grub],[0 0],'r\*', 'LineWidth', 2);  plot([R\_neig R\_teig], [0 0], 'bp', 'LineWidth', 2);  plot(tikslus\_intervalas, f(tikslus\_intervalas), 'k-', 'LineWidth', 2);  title(['f(x)=', f\_name, ' Tikslus intervalas.']);  legend('Grubus šaknų intervalo įvertis', 'Tikslesnis šaknų intervalo įvertis', 'Daugianaris f(x)');  axis([R\_neig R\_teig -25 50]);  plot([-R\_grub, R\_grub], [0, 0], 'b'); % X ašies linija  % ------------------------------------  % g(x)  figure(3); hold on; grid on;  g\_min = -3;  g\_max = 3;  g\_intervalas = g\_min:0.1:g\_max;  plot([g\_min g\_max], [0 0], 'r\*', 'LineWidth', 2);  plot(g\_intervalas, g(g\_intervalas), 'k-', 'LineWidth', 2);  title(['g(x)=', g\_name]);  legend('funkcijos g(x) intervalo rėžiai', 'Funkcija g(x)');  axis([g\_min g\_max -6 2]);  plot([-R\_grub, R\_grub], [0, 0], 'b'); % X ašies linija  end |

|  |
| --- |
| function [R\_grub, R\_neig, R\_teig] = Reziai(n, a)  %Rgrub  R\_grub = 1 + max(abs(a(2:end)))/a(1);  % Rteig skaiciavimas  b = a(2:end);  B = max(abs(b(b<0)));  k = n - (n - (find(b<0, 1)));  R\_teig = 1 + (B/a(1))^(1/k);  % Rneig skaiciavimas  if mod(n, 2) == 0  a(end:-2:1) = -a(end:-2:1);  b = a(2:end);  B = max(abs(b(b<0)));  k = n - (n - (find(b<0, 1)));  R\_neig = 1 + (B/a(1))^(1/k);  R\_neig = -R\_neig;  else  a(end:-2:1) = -a(end:-2:1);  a = a.\*-1;  b = a(2:end);  B = max(abs(b(b<0)));  k = n - (n - (find(b<0, 1)));  R\_neig = 1 + (B/a(1))^(1/k);  R\_neig = -R\_neig;  end;  end |

# Šaknų atskyrimas skenavimo metodu.

## f(x) ir g(x) grafikai ir intervalai

Skenavimas atliekamas intervale [-3.449 ; 4.746], skenavimo žingsnis lygus 0,35. Funkcijos [-3; 3]

A graph with a line

Description automatically generated with medium confidence

pav. Daugianario šaknų atskyrio intervalai.

A graph with lines and dots

Description automatically generated

pav. Funkcijos šaknų atskyrimo intervalai

|  |  |
| --- | --- |
| Intervalas Nr. | Daugianario intervalas |
| 1 | [-1.349489742783177 ; -0.999489742783177] |
| 2 | [-0.649489742783177 ; -0.299489742783177] |
| 3 | [1.100510257216822 ; 1.450510257216822] |
| 4 | [3.900510257216823 ; 4.250510257216823] |

|  |  |
| --- | --- |
| Intervalo Nr. | Funkcijos intervalas |
| 1 | [-2.600000000000000 ; -2.399999999999999] |
| 2 | [-2.199999999999999 ; -1.999999999999999] |
| 3 | [-1.799999999999999 ; -1.599999999999999] |
| 4 | [1.600000000000000 ; 1.800000000000000] |
| 5 | [2.400000000000001 ; 2.600000000000001] |

## Programos kodo fragmentai

|  |
| --- |
| function saknu\_intervalai  clc, close all, clear all;  format long;  % ------------------------------------  % Daugianaris f(x)  f = @(x)(-0.67\*x.^4) + 2.51\*x.^3+2.27\*x.^2-4.02\*x-2.48;  %f = @(x)(0.67\*x.^4) - 2.51\*x.^3-2.27\*x.^2+4.02\*x+2.48;  f\_name = '-0.67x^4+2.51x^3+2.27x^2-4.02x-2.48';  % ------------------------------------  % Funkcija g(x)  g = @(x)exp(-1\*x.^2).\*sin(x.^2).\*(x+2);  g\_name = 'e^-^x\*sin(x^2)\*(x+2)';  % ------------------------------------  a = [0.67 -2.51 -2.27 4.02 2.48];  n = numel(a);  [R\_grub, R\_neig, R\_teig]=Reziai(n, a);  colors = ['g', 'r', 'b', 'k', 'y', 'c'];  % ------------------------------------  % šaknų intervalų atskyrimas daugianariui f(x)  zingsnis = 0.35; % zingsnio nustatymas  [SaknuIntervalai\_fx]=SkenavimasPastoviu(R\_neig, R\_teig, zingsnis, f);  SaknuIntervalai\_fx  % daugianario f(x) ir jo šaknų intervalų atvaizdavimas  figure(1); hold on; grid on;  tikslus\_intervalas = R\_neig:0.1:R\_teig;  plot([R\_neig R\_teig], [0 0], 'bp', 'LineWidth', 2);  plot(tikslus\_intervalas, f(tikslus\_intervalas), 'k-', 'LineWidth', 2);  for i = 1:length(SaknuIntervalai\_fx)  plot(SaknuIntervalai\_fx(i, 1), 0\*SaknuIntervalai\_fx(i, 1), 'o', 'MarkerFaceColor', colors(i), 'MarkerSize', 5);  plot(SaknuIntervalai\_fx(i, 2), 0\*SaknuIntervalai\_fx(i, 2), 'o', 'MarkerFaceColor', colors(i), 'MarkerSize', 5);  end  title(['f(x)=', f\_name, ' Šaknų atskyrimo intervalai. Žingsnis: ', num2str(zingsnis)]);  legend('f(x) šaknų intervalo rėžiai', 'Daugianaris f(x)');  axis([R\_neig R\_teig -25 50]);  % ------------------------------------  % šaknų intervalų atskyrimas funkcijai g(x)  zingsnis = 0.2; % zingsnio nustatymas  g\_min = -3;  g\_max = 3;  g\_intervalas = g\_min:0.1:g\_max;  [SaknuIntervalai\_gx]=SkenavimasPastoviu(g\_min, g\_max, zingsnis, g);  SaknuIntervalai\_gx  figure(2); hold on; grid on;  plot([g\_min g\_max], [0 0], 'rp', 'LineWidth', 2);  plot(g\_intervalas, g(g\_intervalas), 'k-', 'LineWidth', 2);  for i = 1:length(SaknuIntervalai\_gx)  plot(SaknuIntervalai\_gx(i, 1), 0\*SaknuIntervalai\_gx(i, 1), 'o', 'MarkerFaceColor', colors(i), 'MarkerSize', 5);  plot(SaknuIntervalai\_gx(i, 2), 0\*SaknuIntervalai\_gx(i, 2), 'o', 'MarkerFaceColor', colors(i), 'MarkerSize', 5);  end  title(['g(x)=', g\_name, ' Šaknų atskyrimo intervalai. Žingsnis: ', num2str(zingsnis)]);  legend('funkcijos g(x) intervalo rėžiai', 'Funkcija g(x)');  axis([g\_min g\_max -6 2]);  end |

# Šaknų tikslinimas skenavimo, ~~pusiaukirtos~~, Kvazi-Niutono (kirstinių) metodais.

## f(x) daugianaris

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kvazi-Niutono metodas | Pradinis intervalas | Šaknis | Tikslumas | Iteracijų skaičius |
| [-1.349489742783177;  - 0.999489742783177] | -1.180246029145651 | 0.000000000000012 | 7 |
| [-0.649489742783177;  -0.299489742783177] | -0.566365716276059 | 0.000000000001030 | 5 |
| [1.100510257216822; 1.450510257216822] | 1.330278155028645 | 0.000000000006279 | 5 |
| [3.900510257216823; 4.250510257216823] | 4.162602247119908 | 0.000000000453074 | 5 |

A graph with a red line

Description automatically generated

A graph with a red line

Description automatically generated

A graph with a red line

Description automatically generated

A graph with a red line

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Skenavimo metodas | Pradinis intervalas | Šaknis | Tikslumas | Iteracijų skaičius |
| [-1.349489742783177;  - 0.999489742783177] | -1.180246029145651 | 0.000000004758209 | 10 |
| [-0.649489742783177;  -0.299489742783177] | -0.566365716276059 | 0.000000001819280 | 10 |
| [1.100510257216822; 1.450510257216822] | 1.330278155028645 | 0.000000006224549 | 10 |
| [3.900510257216823; 4.250510257216823] | 4.162602247119908 | 0.000000053046186 | 10 |

A graph with a line and a red line

Description automatically generated

A graph with a line graph

Description automatically generated

A graph with lines and numbers

Description automatically generated

A graph with a line and a red line

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MATLAB funkcijos | Pradinis intervalas | Šaknis (fzero) | Šaknis (roots) |  |
| -1.349489742783177 | -1.180246029145653 | -1.180246029145654 |  |
| -0.649489742783177 | -0.566365716276339 | -0.566365716276338 |  |
| 1.100510257216822 | 1.330278155027950 | 1.330278155027949 |  |
| 1. 3.900510257216823 | 4.162602247110461 | 4.162602247110462 |  |

## g(x) funkcija

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kvazi-Niutono metodas | Pradinis intervalas | Šaknis | Tikslumas | Iteracijų skaičius |
| [-2.800000000000000;  -2.700000000000000] | -3.069980129481733 | 0.000000000002991 | 8 |
| [-2.399999999999999;  -2.299999999999999] | -2.506628140055660 | 0.000000000638296 | 4 |
| [-1.999999999999999;  -1.899999999999999] | -2.000000000000000 | 0 | 1 |
| [1.400000000000000; 1.500000000000000] | 1.772453850902849 | 0.000000000001541 | 7 |
| [2.200000000000001; 2.300000000000001] | 2.506628274240725 | 0.000000000016466 | 7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Skenavimo metodas | Pradinis intervalas | Šaknis | Tikslumas | Iteracijų skaičius |
| [-2.600000000000000;  -2.399999999999999] | -2.506628275000000 | 0.000000000002993 | 10 |
| [-2.199999999999999;  -1.999999999999999] | -1.999999999000001 | 0.000000000027723 | 10 |
| [-1.799999999999999;  -1.599999999999999] | -1.772453850999999 | 0.000000000031564 | 10 |
| [1.600000000000000; 1.800000000000000] | 1.772453850999999 | 0.000000000632502 | 10 |
| [2.400000000000001; 2.500000000000001] | 2.506628275000001 | 0.000000000057759 | 10 |

|  |  |  |
| --- | --- | --- |
| MATLAB funkcijos | Pradinis intervalas | Šaknis (fzero) |
| -2.600000000000000 | -2.506628274631001 |
| -2.199999999999999 | -2 |
| 1.600000000000000 | 1.772453850905516 |
| 2.400000000000001 | 2.506628274631001 |

## Programos kodo fragmentai

|  |
| --- |
| function saknu\_tikslinimas  clc, close all, clear all;  format long;  % ------------------------------------  % Daugianaris f(x)  f = @(x)(-0.67\*x.^4) + 2.51\*x.^3+2.27\*x.^2-4.02\*x-2.48;  %f = @(x)(0.67\*x.^4) - 2.51\*x.^3-2.27\*x.^2+4.02\*x+2.48;  f\_name = '-0.67x^4+2.51x^3+2.27x^2-4.02x-2.48';  % ------------------------------------  % Funkcija g(x)  g = @(x)exp(-1\*x.^2).\*sin(x.^2).\*(x+2);  g\_name = 'e^-^x\*sin(x^2)\*(x+2)';  % ------------------------------------  a = [0.67 -2.51 -2.27 4.02 2.48];  n = numel(a);  [R\_grub, R\_neig, R\_teig]=Reziai(n, a);  % ------------------------------------  % šaknų intervalų atskyrimas daugianariui f(x)  % ------------------------------------  zingsnis = 0.35; % zingsnio nustatymas  [SaknuIntervalai\_fx]=SkenavimasPastoviu(R\_neig, R\_teig, zingsnis, f);  % ------------------------------------  % REKURSINIS SKENAVIMAS (MAŽINANT ŽINGSNĮ)  % ------------------------------------  % šaknų tikslinimas daugianariui f(x)  % ------------------------------------  fprintf( '----------------------------------------------------\n');  fprintf( 'Šaknų tikslinimas skenavimo metodu, mažinant žingsnį\n');  fprintf( 'Daugianaris f(x)=-0.67x^4+2.51x^3+2.27x^2-4.02x-2.48\n');  fprintf( '----------------------------------------------------\n');  fprintf( 'Stulpelių reikšmės:\n');  fprintf( '1:2 - pradiniai šaknų tikslinimo intervalai\n');  fprintf( '3 - šaknis\n');  fprintf( '4 - tikslumas\n');  fprintf( '5 - atliktų iteracijų kiekis\n');  fprintf( '----------------------------------------------------\n');  Saknys\_intervalai\_fx = [];  Tikslumai = [];  Iteracijos = [];  tikslumas = 1e-9;  for i=1:length(SaknuIntervalai\_fx)  x\_min = SaknuIntervalai\_fx(i,1);  x\_max = SaknuIntervalai\_fx(i, 2);  if i == 1  draw = 1;  figure(1); grid on; hold on;  npoints= 1000;  x = x\_min:(x\_max-x\_min)/(npoints - 1):x\_max;  plot(x, f(x), 'r-', 'LineWidth', 2);  else  draw = 0;  end;  if (sign(f(x\_min)) ~= sign(f(x\_max)))  iteracijos\_sk = 0;  [a, b, it, t]=SkenavimasRekursija(x\_min, x\_max, zingsnis, tikslumas, f, iteracijos\_sk, draw);  Saknys\_intervalai\_fx = [Saknys\_intervalai\_fx; a b];  Iteracijos = [Iteracijos; it];  Tikslumai = [Tikslumai; t];  end  end  close all;  Saknys\_fx = (Saknys\_intervalai\_fx(:,1) + Saknys\_intervalai\_fx(:,2))/2;  Rez\_fx = [];  for i=1:length(Saknys\_fx)  Rez\_fx = [Rez\_fx; SaknuIntervalai\_fx(i,:) Saknys\_fx(i) Tikslumai(i) Iteracijos(i)];  end  Rez\_fx  % ------------------------------------  % šaknų tikslinimas funkcijai g(x)  % ------------------------------------  % šaknų intervalų atskyrimas funkcijai g(x)  % ------------------------------------  zingsnis = 0.2; % zingsnio nustatymas  g\_min = -3;  g\_max = 3;  [SaknuIntervalai\_gx]=SkenavimasPastoviu(g\_min, g\_max, zingsnis, g);  fprintf( '----------------------------------------------------\n');  fprintf( 'Šaknų tikslinimas skenavimo metodu, mažinant žingsnį\n');  fprintf( 'Funkcija g(x)=e^-^x\*sin(x^2)\*(x+2)\n');  fprintf( '----------------------------------------------------\n');  fprintf( 'Stulpelių reikšmės:\n');  fprintf( '1:2 - pradiniai šaknų tikslinimo intervalai\n');  fprintf( '3 - šaknis\n');  fprintf( '4 - tikslumas\n');  fprintf( '5 - atliktų iteracijų kiekis\n');  fprintf( '----------------------------------------------------\n');  Saknys\_intervalai\_gx = [];  Tikslumai = [];  Iteracijos = [];  tikslumas = 1e-9;  for i=1:length(SaknuIntervalai\_gx)  x\_min = SaknuIntervalai\_gx(i, 1);  x\_max = SaknuIntervalai\_gx(i, 2);  draw = 0;  if (sign(g(x\_min)) ~= sign(g(x\_max)))  iteracijos\_sk = 0;  [a, b, it, t]=SkenavimasRekursija(x\_min, x\_max, zingsnis, tikslumas, g, iteracijos\_sk, draw);  Saknys\_intervalai\_gx = [Saknys\_intervalai\_gx; a b];  Iteracijos = [Iteracijos; it];  Tikslumai = [Tikslumai; t];  end  end  Saknys\_gx = (Saknys\_intervalai\_gx(:,1) + Saknys\_intervalai\_gx(:,2))/2;  Rez\_gx = [];  for i=1:length(Saknys\_gx)  Rez\_gx = [Rez\_gx; SaknuIntervalai\_gx(i,:) Saknys\_gx(i) Tikslumai(i) Iteracijos(i)];  end  Rez\_gx  % ------------------------------------  % Kvazi-Niutono (kirstinių) metodas  eps = 1e-9;  % ------------------------------------  % šaknų tikslinimas daugianariui f(x)  % ------------------------------------  fprintf( '----------------------------------------------------\n');  fprintf( 'Šaknų tikslinimas Kvazi-Niutono (kirstinių) metodu\n');  fprintf( 'Daugianaris f(x)=-0.67x^4+2.51x^3+2.27x^2-4.02x-2.48\n');  fprintf( '----------------------------------------------------\n');  fprintf( 'Stulpelių reikšmės:\n');  fprintf( '1 - pirmasis pradinis artinys\n');  fprintf( '2 - antrasis pradinis artinys\n');  fprintf( '3 - šaknis\n');  fprintf( '4 - tikslumas\n');  fprintf( '5 - atliktų iteracijų kiekis\n');  fprintf( '----------------------------------------------------\n');  Tikslumai = [];  Iteracijos = [];  Saknys\_fx = [];  Artiniai = [];  iteracijos\_sk\_max = 200;  figure(1); grid on; hold on;  for i=1:length(SaknuIntervalai\_fx)  x0 = SaknuIntervalai\_fx(i, 1);  x01 = SaknuIntervalai\_fx(i, 2);  npoints=1000;  x=x0:(x01-x0)/(npoints-1):x01;  axis([(x0-0.01) (x01+0.01) -2.5 8.5]);  Artiniai = [Artiniai; x0 x01];  fxn = f(x0);  fxn1 = f(x01);  xn = x0;  xn\_plot = x0;  xn1\_plot = x01;  fxn\_plot = f(x0);  fxn1\_plot = f(x01);  dfxn = (fxn1 - fxn)/(x01-x0);  tikslumas = 1;  iteracijos\_sk = 0;  while tikslumas > eps  iteracijos\_sk = iteracijos\_sk + 1;  if (iteracijos\_sk > iteracijos\_sk\_max)  fprintf('Virsytas leistinas iteraciju skaicius');  break;  end  xn1 = xn - fxn/dfxn;  if(i == 1 && iteracijos\_sk < 7)  plot(x,f(x),'r-');  plot([x0 x01],[0 0],'b-');  plot(x0,0,'mp');  h = findobj(gca,'Type','line');h1=h(1);  plot([xn\_plot,xn\_plot,xn1\_plot,xn1\_plot],[0,fxn\_plot,fxn1\_plot,0],'k-');  plot([xn,xn,xn1],[0,fxn,0],'k-');  delete(h1);plot(xn1,0,'mp');h = findobj(gca,'Type','line');h1=h(1);  input('Press Enter'), figure(1);  end  fxn1 = f(xn1);  dfxn = (fxn1 - fxn)/(xn1 - xn);  xn = xn1;  fxn = f(xn);  tikslumas = abs(fxn);  end  Iteracijos = [Iteracijos; iteracijos\_sk];  Tikslumai = [Tikslumai; tikslumas];  Saknys\_fx = [Saknys\_fx; xn];  end;  close all;  Rez\_fx = [];  for i=1:length(Saknys\_fx)  Rez\_fx = [Rez\_fx; Artiniai(i, :) Saknys\_fx(i) Tikslumai(i) Iteracijos(i)];  end;  Rez\_fx  % ------------------------------------  % šaknų tikslinimas funkcijai g(x)  % ------------------------------------  fprintf( '----------------------------------------------------\n');  fprintf( 'Šaknų tikslinimas Kvazi-Niutono (kirstinių) metodu\n');  fprintf( 'Funkcija g(x)=sin(x)ln(x)-(x/6)\n');  fprintf( '----------------------------------------------------\n');  fprintf( 'Stulpelių reikšmės:\n');  fprintf( '1 - pirmasis pradinis artinys\n');  fprintf( '2 - antrasis pradinis artinys\n');  fprintf( '3 - šaknis\n');  fprintf( '4 - tikslumas\n');  fprintf( '5 - atliktų iteracijų kiekis\n');  fprintf( '----------------------------------------------------\n');  Tikslumai = [];  Iteracijos = [];  Saknys\_gx = [];  Artiniai = [];  iteracijos\_sk\_max = 200;  for i=1:length(SaknuIntervalai\_gx)  x0 = SaknuIntervalai\_gx(i, 1) - 0.2;  x01 = SaknuIntervalai\_gx(i, 1) - 0.1;  Artiniai = [Artiniai; x0 x01];  fxn = g(x0);  fxn1 = g(x01);  dfxn = (fxn1 - fxn)/(x01-x0);  xn = x0;  tikslumas = 1;  iteracijos\_sk = 0;  while tikslumas > eps  iteracijos\_sk = iteracijos\_sk + 1;  if (iteracijos\_sk > iteracijos\_sk\_max)  fprintf('Virsytas leistinas iteraciju skaicius');  break;  end  xn1 = xn - fxn/dfxn;  fxn1 = g(xn1);  dfxn = (fxn1 - fxn)/(xn1 - xn);  xn = xn1;  fxn = g(xn);  tikslumas = abs(fxn);  end  Iteracijos = [Iteracijos; iteracijos\_sk];  Tikslumai = [Tikslumai; tikslumas];  Saknys\_gx = [Saknys\_gx; xn];  end;  Rez\_gx = [];  for i=1:length(Saknys\_gx)  Rez\_gx = [Rez\_gx; Artiniai(i, :) Saknys\_gx(i) Tikslumai(i) Iteracijos(i)];  end;  Rez\_gx  % Matlab funkcijos  % Daugianaris f(x)  fprintf( '----------------------------------------------------\n');  fprintf( 'MATLAB funkcijos\n');  fprintf( '----------------------------------------------------\n');  fprintf( 'Daugianaris f(x)\n');  fprintf( '----------------------------------------------------\n');  a = [0.67 -2.51 -2.27 4.02 2.48];  saknys\_roots = roots(a)  for i=1:length(SaknuIntervalai\_fx)  fzero(f, SaknuIntervalai\_fx(i, 1))  end  % Matlab funkcijos  % Funkcija g(x)  fprintf( '----------------------------------------------------\n');  fprintf( 'Funkcija g(x)\n');  fprintf( '----------------------------------------------------\n');  for i=1:length(SaknuIntervalai\_gx)  fzero(g, SaknuIntervalai\_gx(i, 1))  end  end |

# 2 dalis

## A graph of a function Description automatically generatedg(x) ir TE tarpiniai grafikai

6 pav. g(x) ir TE tarpiniai grafikai

## A graph with lines and numbers Description automatically generatedg(x) ir TE grafikai su reikalaujančiu tikslumu

7 pav. g(x) ir TE grafikai su reikalaujančiu tikslumu

## TE analitinė išraiška

TE analitinė išraiška daugianario pavidalu N=5: 6.96745992622116\*x + 0.550885669585957\*(0.666666666666667\*x + 1)^4 - 7.26042088452591\*(0.666666666666667\*x + 1)^3 - 1.03293679565023\*(0.666666666666667\*x + 1)^2 + 12.9459959057301

## A graph with a line and numbers Description automatically generatedSprendinai su skirtingu TE narių skaičiumi

pav. Šaknų skaičius pagal TE

A graph with a line and numbers

Description automatically generated

9 pav. Pirmos šaknies tikslumas

A graph with a line and numbers

Description automatically generated

10 pav. Antros šaknies tikslumas

A graph with a line and numbers

Description automatically generated

11 pav. Trečios šaknies tikslumas

## Programos kodo fragmentai

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import sympy as sp  # Declare symbols  x = sp.symbols("x")  # Declare function using the given format  def fh(x):  return 5\*sp.cos(x) - sp.sin(2\*x) + 2  # Taylor method  def Taylor(f, x, x0, N):  f\_sympy = f(x) # Use function call to get the sympy expression  fp = f\_sympy.subs(x, x0) # Gets the value of the function  df = f\_sympy  for i in range(1, N+1):  df = df.diff(x) # Differentiating  fp = fp + df.subs(x, x0) \* (x - x0)\*\*i / sp.factorial(i) # Computes the Taylor series  return fp  # Newton's method  def newton(f, df, x0, max\_iter=1000):  x\_current = x0  for \_ in range(max\_iter):  # Calculates functional values for the function and its derivative with the current x value  f\_val = f(x\_current)  df\_val = df.subs(x, x\_current)  if df\_val == 0:  raise ValueError("Derivative is zero!")  dx = -f\_val / df\_val # Compute change Δx  if abs(dx) < 1e-4: # Check if change is small enough  return x\_current  x\_current = x\_current + dx  raise ValueError(f"Newton's method did not converge after {max\_iter} iterations")  def find\_roots(func, df, start, end, step=0.1):  # Scanning for sign changes  intervals = []  x\_val = start  while x\_val < end:  if func(x\_val) \* func(x\_val + step) <= 0:  intervals.append((x\_val, x\_val + step))  x\_val += step  # Applying Newton's method for detected intervals  roots = []  for region in intervals:  x0 = (region[0] + region[1]) / 2  root = newton(func, df, x0)  if root not in roots:  roots.append(root)  return roots  a = -6  b = 3  x0\_val = (a + b) / 2  # Find the derivative of h(x)  fh\_der = fh(x).diff(x)  # Find all roots for h(x)  roots\_hx = find\_roots(fh, fh\_der, a, b)  print(f"h(x) šaknys: {roots\_hx}")  # Find all roots for the Taylor expansion  N = 1  while True:  f\_N = Taylor(fh, x, x0\_val, N)  f\_N\_der = f\_N.diff(x)  roots\_f\_N = find\_roots(lambda x\_val: f\_N.subs(x, x\_val), f\_N\_der, a, b)  # Compare roots of f\_N with roots of h(x) with given accuracy  roots\_match = True  for root\_f\_N in roots\_f\_N:  is\_match\_for\_this\_root = False  for root\_hx in roots\_hx:  if abs(root\_f\_N - root\_hx) < 1e-4:  is\_match\_for\_this\_root = True  break  if not is\_match\_for\_this\_root:  roots\_match = False  break  if roots\_match:  break  N += 1  print(f"Kad Teiloro eilutė pasiektų visas šaknis, reikia {N} narių")  print(f"Teiloro eilutės šaknys su N={N}: {roots\_f\_N}")  print(f"h(x) šaknys: {roots\_hx}")  print("\nPirma užduotis:\n")  f\_3 = Taylor(fh, x, x0\_val, 3)  f\_4 = Taylor(fh, x, x0\_val, 4)  f\_5 = Taylor(fh, x, x0\_val, 5)  xxx = np.linspace(a, b, 200)  hx\_values = [fh(x).subs(x, val) for val in xxx]  f3\_values = [f\_3.subs(x, val) for val in xxx]  f4\_values = [f\_4.subs(x, val) for val in xxx]  f5\_values = [f\_5.subs(x, val) for val in xxx]  plt.figure()  plt.ylim(-500, 500)  plt.plot([a, b], [0, 0], 'k--')  plt.plot(xxx, hx\_values, label='h(x)', color='orange')  plt.plot(xxx, f3\_values, label='TE, N=3', color='blue', linestyle='dashed')  plt.plot(xxx, f4\_values, label='TE, N=4', color='green', linestyle='dashed')  plt.plot(xxx, f5\_values, label='TE, N=5', color='red', linestyle='dashed')  plt.xlabel('x')  plt.ylabel('y')  plt.plot(roots\_hx, np.zeros(len(roots\_hx)), 'mp')  plt.legend()  plt.grid(True)  plt.title("g(x) ir TE tarpiniai grafikai")  plt.show()  print("\nAntra užduotis:\n")  fp = Taylor(fh, x, x0\_val, N)  npoints = 200  xxx = np.linspace(a, b, npoints)  ggg = []  tggg = []  for i in range(npoints):  ggg.append(fh(x).subs(x, xxx[i]))  tggg.append(fp.subs(x, xxx[i]))  plt.figure()  plt.plot(xxx, ggg, 'b-', label='g(x)')  plt.plot(xxx, tggg, 'r-', label=f'Teiloro eilute, N={N}')  plt.plot([a, b], [0, 0], 'k--')  plt.grid()  plt.xlabel('x')  plt.ylabel('g(x), TE')  plt.legend()  plt.title("g(x) ir TE grafikai su reikalaujančiu tikslumu")  plt.plot(roots\_hx, np.zeros(len(roots\_hx)), 'mp')  plt.show()  print("\nTrečia užduotis:\n")  taylor\_expansion = fh(x).series(x, x0\_val, N).removeO()  print(f"TE analitinė išraiška daugianario pavidalu N={N}: {taylor\_expansion}")  print("\nKetvirta užduotis:\n")  # For part a)  TE\_terms = list(range(1, N+1))  root\_counts = []  for i in TE\_terms:  f\_i = Taylor(fh, x, x0\_val, i)  f\_i\_der = f\_i.diff(x)  roots\_f\_i = find\_roots(lambda x\_val: f\_i.subs(x, x\_val), f\_i\_der, a, b)  root\_counts.append(len(roots\_f\_i))  plt.figure()  plt.plot(TE\_terms, root\_counts, '-o')  plt.xlabel('TE')  plt.ylabel('Šaknų skaičius')  plt.title('Šaknų skaičius pagal Teiloro eilutę')  plt.grid(True)  plt.show()  # For part b)  root\_index = 1  for root\_hx in roots\_hx:  errors = []  for i in TE\_terms:  f\_i = Taylor(fh, x, x0\_val, i)  f\_i\_der = f\_i.diff(x)  roots\_f\_i = find\_roots(lambda x\_val: f\_i.subs(x, x\_val), f\_i\_der, a, b)  nearest\_root = min(roots\_f\_i, key=lambda root: abs(root-root\_hx))  error = abs(nearest\_root - root\_hx)  errors.append(error)    if len(TE\_terms) == len(errors): # Check if lengths match  plt.figure()  plt.plot(TE\_terms, errors, '-o')  plt.xlabel('TE')  plt.ylabel('Tikslumas')  plt.title(f'Tikslumas šakniai nr. {root\_index}: {root\_hx}')  plt.grid(True)  plt.yscale('log')  plt.show()  root\_index += 1 |