informacijos
sistemų
katedra

KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas

P170B115 Skaitiniai metodai ir algoritmai

Laboratorinis darbas Nr. 1

Variantas 1

Dėstytojas:
prof. BARAUSKAS Rimantas
Studentė:
Laura Capaitė IFK-0

KAUNAS, 2022

Turinys

1. Įvadas

Šio laboratorinio darbo esmė išmokti skaičiuoti sudėtingų lygčių nežinomuosius sprendinius pasinaudojant kompiuteriu. Išmokti paskaičiuoti grubius bei tiksliuosius intervalus. Naudojant skenavimą nekintančiu žingsniu rasti šaknų intervalus, pritaikyti įvairius tikslinimo metodus.

2. Užduotis

Pirmo varianto užduotis (1 pav.)

Varianto Nr.	Daugianariai $f(x)$	Funkcijos $g(x)$	Metodai[1]
1	$0.10x^5 - 0.05x^4 - 1.95x^3 + 1.75x^2 + 5.18x - 2.14$	$\frac{(x+1)^2(x-3)^2}{x^3+2} + (x-2)^3\cos(x); 0 \le x \le 15$	2, 3, 5

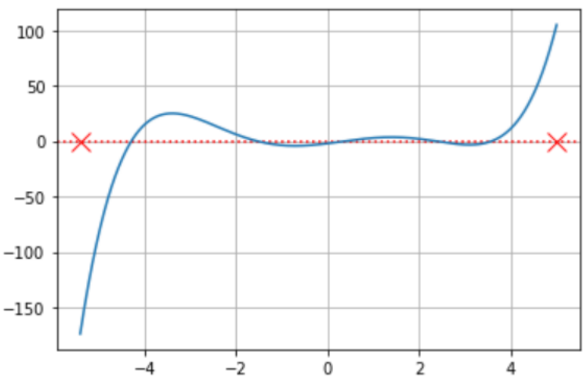
(1 pav. užduoties funkcijos)

Funkcijų sprendimui naudojami trys metodai iš metodų lentelės (2 pav.): paprastųjų iteracijų, Niutono (liestinių) ir skenavimo mažėjančiu žingsniu.

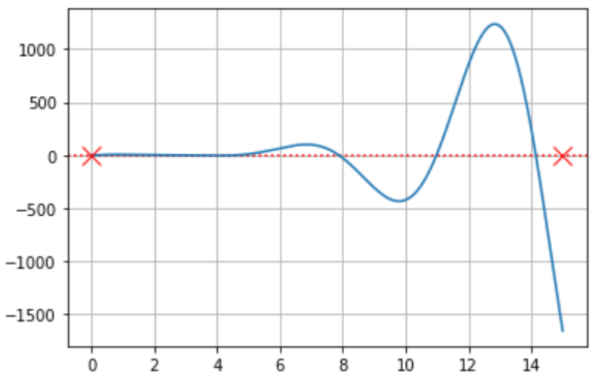
Metodo Nr.	Metodo pavadinimas
1	Stygų
2	Paprastųjų iteracijų
3	Niutono (liestinių)
4	Kvazi-Niutono (kirstinių)
5	Skenavimo su mažėjančiu žingsniu

(2 pav. metodų lentelė)

2.1. F(x) ir g(x) grafikai ir šaknų intervalų galų apskaičiavimas



(3 pav. f(x) funkcijos grafikas intervale [-5, 2])



(4 pav. g(x) funkcijos grafikas intervale [0, 15])

Naudojant „grubų“ šaknų įvertinimą f(x) funkcijai gaunamas intervalas:
`[-52.8, 52.8]`

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ \dots \ + \ a_1 x + a_0 = 0, \qquad\qquad a_n > 0$$

$$|x| < 1 + \frac{\max\limits_{0 \le i \le n-1} |a_i|}{a_n} = R$$

max|a_i| = 5.18
a_5 = 0.10
R = 1 + 5.18/0.10 = 52.8
[-52.8, 52.8]

Naudojant „tikslesnį“ šaknų įvertinimą f(x) funkcijai gaunamas intervalas:
`[-3,69, 2.845]`

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ \dots \ + \ a_1 x + a_0 = 0, \qquad a_n > 0$$

Teigiamoms šaknims

$$x \le R_{teig}, \ R_{teig} = 1 + \sqrt[k]{\frac{B}{a_n}}, \quad k = n - \max_{0 \le i \le n-1}(i, a_i < 0), \quad B = \max_{0 \le i \le n-1}(|a_i|, a_i < 0)$$

„vyriausio“ neigiamo koeficiento numeris

Absoliutine reikšme didžiausio neigiamo koeficiento absoliutinė reikšmė

B = 2.14
k = 5 – 0 = 5
$R_{teig} = 1 + \sqrt[5]{\frac{2.14}{0.10}} = 2.845$

B = 1.95
k = 5 – 2 = 3
$R_{neig} = 1 + \sqrt[3]{\frac{1.95}{0.1}} = \ 3.69$

-min(R,R_{neig})<=x<=min(R,R_{teig})

-min(52.8, 3.69)<=x<=min(52.8, 2.845)
[-3.69, 2.845]

2.2. Šaknų intervalų suradimas skenavimo algoritmu su nekintančiu skenavimo žingsniu

```
def scan(interval, func):
  x1 = interval[0];
  x2 = x1 + STEP;
  root_intervals = [];
  while x1 < interval[1]:
    if (np.sign(func(x1)) != np.sign(func(x2))):
      root_intervals.append([x1, x2])
    x1 = x1 + STEP
    x2 = x2 + STEP
  return root_intervals;
```

(5 pav. skenavimo algoritmo su nekintančiu skenavimo žingsniu kodas)

Gauti šaknų intervalai f(x) ir g(x) funkcijose yra nurodyti lentelėse kartu su rezultatais.

2.3. Šaknų suradimas paprastųjų iteracijų metodu

```
def simple_iteration(interval, func, i):
  x = interval[0] ;
  precision = 1;
  iterations = 0;
  while precision > EPS and maxI > iterations:
    iterations += 1;
    x_next = (func(x)/alpha[i]) + x;

    precision = abs(x - x_next)
    x = x_next
  return x, iterations, precision
```

(6 pav. paprastųjų iteracijų metodo kodas)

Kiekvienam intervalui yra parinkta atitinkama alpha reikšmė, kad šaknies suradimui iteracijų skaičius būtų kuo mažesnis.

f(x) funkcijos rezultatai(7 pav.):

Parastųjų iteracijų metodas

Intervalas	Šaknis	Iteracijos	Tikslumas	Alpha
-4.283999999999628 - -4.282999999999627	-4.283488499352873	15	6.478639846818623e-11	-100
-1.5399999999998042 - -1.5389999999998043	-1.5392768458479302	7	1.7392087769962927e-11	10
0.3840000000001376 - 0.3850000000001376	0.3846181776362378	20	4.524614016787609e-11	-10
2.354999999999989 - 2.3559999999999888	2.355222610763822	16	5.566214156260685e-11	10
3.5819999999998537 - 3.5829999999998536	3.5829245566736283	32	7.58832996439196e-11	-10

(7 pav. f(x) funkcijos rezultatai paprastųjų iteracijų metodu)

g(x) funkcijos rezultatai(8 pav.):

Parastųjų iteracijų metodas

Intervalas	Šaknis	Iteracijos	Tikslumas	Alpha
0.20500000000000015 - 0.20600000000000016	0.20596933334762108	23	4.875130854564702e-11	-30
2.5489999999998303 - 2.54999999999983	2.5495022628415045	26	6.838352106797174e-11	1
4.665999999999893 - 4.666999999999893	4.666986057111641	7	4.958256027975949e-11	-20
7.872000000000964 - 7.873000000000964	7.872818235303256	5	7.524203482489611e-11	200
10.98599999999935 - 10.98699999999935	10.986065268342262	10	5.5296212053690397e-11	-600
14.141999999997601 - 14.1429999999976	14.142784332256198	72	8.960121533618803e-11	1000

(8 pav. g(x) funkcijos rezultatai paprastųjų iteracijų metodu)

2.4. Šaknų suradimas Niutono (liestinių) metodu

```
def newton(interval, fx, df, func):
  precision = 1;
  iterations = 0;
  x0 = interval[0];
  while precision > EPS and maxI > iterations:
    iterations += 1;
    x1 = x0 - fx.subs(x,x0).evalf()/df.subs(x,x0).evalf()
    precision = abs(x1-x0)
    x0 = x1
  return x1, iterations, precision;
```

(9 pav. Niutono (liestinių) metodo kodas)

f(x) funkcijos rezultatai(10 pav.):

Niutono liestinių metodas

Intervalas	Šaknis	Iteracijos	Tikslumas
-4.283999999999628 - -4.282999999999627	-4.28348849932082	3	4.17443857259059E-14
-1.5399999999998042 - -1.5389999999998043	-1.53927684584881	3	6.12843109593086E-14
0.3840000000001376 - 0.3850000000001376	0.384618177670927	3	5.55111512312578E-17
2.354999999999989 - 2.3559999999999888	2.35522261079715	3	1.33226762955019E-15
3.5819999999998537 - 3.5829999999998536	3.58292455670155	3	2.23288054712611E-12

(10 pav. f(x) funkcijos rezultatai Niutono (liestinių) metodu)

g(x) funkcijos rezultatai(11 pav.):

Niutono liestinių metodas

Intervalas	Šaknis	Iteracijos	Tikslumas
0.20500000000000015 - 0.20600000000000016	0.205969333392610	3	1.45550238528358E-13
2.5489999999998303 - 2.54999999999983	2.54950226286497	3	9.32587340685131E-15
4.665999999999893 - 4.666999999999893	4.66698605711487	3	1.41842093626110E-12
7.872000000000964 - 7.873000000000964	7.87281823530197	3	5.68434188608080E-14
10.98599999999935 - 10.98699999999935	10.9860652683518	3	0
14.141999999997601 - 14.1429999999976	14.1427843322958	3	5.32907051820075E-15

(11pav. g(x) funkcijos rezultatai Niutono (liestinių) metodu)

2.5. Šaknų suradimas skenavimo algoritmu su kintančiu skenavimo žingsniu

```
def scan_method(root_interval, func):
  scan_step = STEP;
  iterations = 0;
  x1 = root_interval[0];
  x2 = root_interval[1];
  precision = 1;
  while precision > EPS and maxI > iterations:
    iterations += 1;
    if (np.sign(func(x1)) != np.sign(func(x2))):
        scan_step = (scan_step / 2);
        x1 -= scan_step;
        x1 = x1 + scan_step;
        x2 = x1 + scan_step;
    elif func(x1) is 0:
      return x1
    else:
      x1 = x1 + scan_step;
      x2 = x2 + scan_step;
    precision = abs(func(x1))
  return x1, precision, iterations
```

(12 pav. skenavimo algoritmo su kintančiu skenavimo žingsniu kodas)

f(x) funkcijos rezultatai(13 pav.):

Skenavimo mažėjančiu žingsniu metodas

Intervalas	Šaknis	Iteracijos	Tikslumas
-4.283999999999628 - -4.282999999999627	-4.283488499321602	46	5.2526427651855556e-11
-1.5399999999998042 - -1.5389999999998043	-1.5392768458573054	36	8.948708440925657e-11
0.3840000000001376 - 0.3850000000001376	0.3846181776673977	37	1.9977797194314917e-11
2.354999999999989 - 2.3559999999999888	2.355222610786546	39	6.63207266882182e-11
3.5819999999998537 - 3.5829999999998536	3.582924556698504	49	4.8181458822682544e-11

(13 pav. f(x) funkcijos rezultatai skenavimo algoritmu su kintančiu skenavimo žingsniu)

f(x) funkcijos rezultatai(14 pav.):

Skenavimo mažėjančiu žingsniu metodas

Intervalas	Šaknis	Iteracijos	Tikslumas
0.20500000000000015 - 0.20600000000000016	0.20596933338791146	42	7.330047679943164e-11
2.5489999999998303 - 2.54999999999983	2.549502262830565	29	5.238437461585477e-11
4.665999999999893 - 4.666999999999893	4.666986057110024	44	9.10685971078351e-11
7.872000000000964 - 7.873000000000964	7.872818235301911	50	1.1093792551264414e-11
10.98599999999935 - 10.98699999999935	10.986065268351744	49	4.6060932845648495e-12
14.141999999997601 - 14.1429999999976	14.142784332295752	46	7.643485844255338e-11

(14 pav. g(x) funkcijos rezultatai skenavimo algoritmu su kintančiu skenavimo žingsniu)

2.6. Šaknys surastos pagal *wolframalpha.com*

$x \approx -4.28349$

$x \approx -1.53928$

$x \approx 0.384618$

$x \approx 2.35522$

$x \approx 3.58292$

(15 pav. f(x) funcijos šaknys)

$x \approx 0.205969333392610\ldots$

$x \approx 2.54950226286497\ldots$

$x \approx 4.66698605711487\ldots$

$x \approx 7.87281823530197\ldots$

(16 pav. g(x) funcijos šaknys)

Funkcijų šaknys surastos naudojant užduotyje nurodytus metodus atitinka gautus pagal *wolframalpha.com*. Tačiau dėl negalėjimo pasirinkti funkcijos intervalo, visos šaknys, užduotuje nurodytame intervale, nebuvo surastos. Tačiau pagal turimas keturias teisingas šaknis, galima teigti, kad ir kitos dvi bus teisingos.

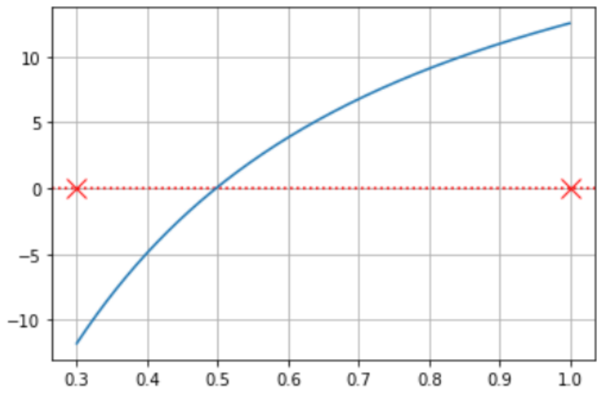
3. Netiesinės lygties sprendimas

Vertikaliai į viršų iššauto objekto greitis užrašomas dėsniu $v(t) = v_0 e^{-\frac{ct}{m}} + \frac{mg}{c}\left(e^{-\frac{ct}{m}} - 1\right)$, čia $g = 9{,}8\ \mathrm{m/s^2}$, pasipriešinimo koeficientas c, pradinis greitis v_0. Kokia objekto masė, jeigu žinoma, kad laiko momentu t_1 objekto greitis buvo lygus v_1?

Varianto Nr.	v_0, m/s	c, kg/s	t_1, s	v_1, m/s
1	100	0,05	5	22

(17 pav. užduotis)

Funcijos grafikas pavaizduotas 18 pav. Intervalas parinktas labai mažas `[0.3, 1]`, kad būtų matoma, kur yra kertama y ašis. Funkcijai spręsti parinktas Niutono (liestinių) metodas (19 pav.), nes jis yra greitesnis, sprendimo gavimui reikalingas mažiausias iteracijų skaičius.



(18 pav. v(x) funkcijos grafikas)

v(x) funcijos rezultatas(19 pav.):

```
Niutono liestinių metodas
|                 Intervalas                  |        Šaknis         | Iteracijos |         Tikslumas          |
| 0.49700000000000016 - 0.49800000000000016  |  0.497840875824689    |      3     |   1.45827794284514E-12     |
```

(19 pav. rezultatai naudojant Niutono (liestinių) metodą)

Objekto masė lygi: 0,497840875824689

4. Išvados

Naudojant užduotyje nurodytus metodus buvo sėkmingai surastos funkcijų šaknys. Niutono (liestinių) metodas veikė greičiausiai, gauti šaknies rezultatą prireikia tik keletos iteracijų. Tuo tarpu skenavimo su mažėjančiu žingsniu metodas užtruko daugiausiai iteracijų.