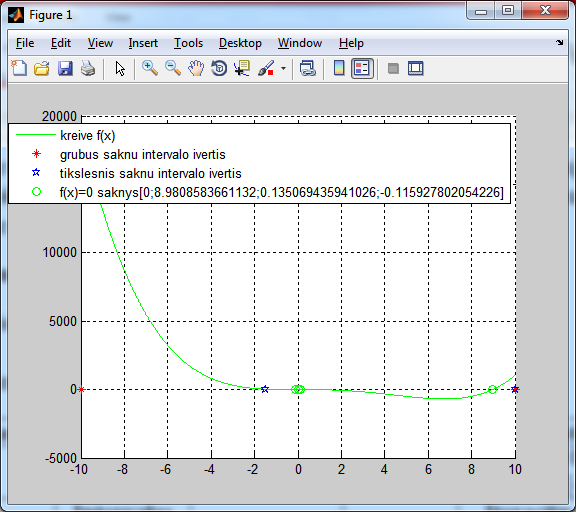
**Kolokviumo užduotys**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Grupė** | **Pavardė Vardas** | **Savarankiško darbo Nr.** | **Lygčių Nr.** | **Sprendimo metodas** | |
| **daugianario** | **sistemos** |
| IF-0/8  IFF–0 | Čerkus Dovydas | 11 | 2 | Niutono | Paprastųjų Iteracijų |

1. *Vienos lygties sprendimo algoritmai*. **Duota** daugianario lygtis f(x)=0.
   1. Grafiškai pavaizduokite funkciją.
   2. Nustatykite daugianario f(x) šaknų intervalą, taikydami „grubų“ ir tikslesnį įverčius.
   3. Apskaičiuokite ir grafiškai pavaizduokite daugianario šaknis, taikydami nurodytą metodą.
   4. Patikrinkite gautas šaknų reikšmes naudodami MATLAB funkciją ***roots***.

Darbo **rezultatus** pateikite *Word* faile, kurį pavadinkite savo pavarde.

1. Daugianario grafiką su pažymėtomis šaknimis.



1. Rezultatų lentelę.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Metodas** | | **Lygtis** | | | **Šaknų intervalo įverčiai** | |
| Niutono | | = 0 | | | Grubus [-10; 10]  Tikslesnis [-1.52002; 10] | |
| **Šaknis** | **Intervalas**  ***arba* artinys** | | **Tikslumas** | **Iteracijų skaičius** | **Iteracijų pabaigos sąlygos** | ***roots* funkcijos rezultatas** |
| 1.05195e-019 | 0,01 | | 9.46758e-019 | 4 |  | 0 |
| 8.98086 | 10 | | 8.09735e-011 | 6 |  | 8.9808583661132 |
| 0.135069 | 4 | | 1.86509e-011 | 13 |  | 0.135069435941026 |
| -0.11592 | -1 | | 1.44995e-013 | 11 |  | - 0.115927802054226 |

1. Programos kodą.

***Failas DaugianarioPavaizdavimas.m***

function DaugianarioPavaizdavimas

clc, close all;

syms f x;

% ------------------------ Daugianaris---------------------------------

f = 64 \* x^4 - 576 \* x^3 + 10 \* x^2 + 9 \* x; % simboline israiska, argumentas x

xRange = [-40; 45];

points = 1000;

figureNo = 1;

x = [xRange(1) : (xRange(2) - xRange(1)) / points : xRange(2)];

figure(1); grid on; hold on;

plot(x, eval(f), 'b-');

fRoots = solve(f, x); % daugianario šaknys

plot(fRoots, 0 \* fRoots, 'go');

legend('kreive f(x)', 'f(x)=0 saknys');

title([char(f), '=0'])

end

***Failas DaugianarioSaknuIverciai.m***

function DaugianarioSaknuIverciai

clc, close all;

syms f x;

f = 64 \* x^4 - 576 \* x^3 + 10 \* x^2 + 9 \* x; % simboline israiska, argumentas x

%---------------------------- Koef skaičiavimas----------------------------

f = expand(f); % daugianario skleistine

fNeg = expand(subs(f,x,-x)); % daugianario skleistine pakeitus x-> -x

[coeffs1, orders] = coeffs(f,x); % coeffs - daugianario koeficientai nuo didžiausio x laipsnio iki mažiausio;

% orders - tik x-sai su savo laipsniais

[coeffs1\_neg, orders\_neg] = coeffs(fNeg, x); % daugianario fNeg koeficientai ir juos atitinkantys x laipsniai

maxXOrder = char( orders(1)); % paverčiam į string

charPos = strfind(maxXOrder, '^'); % ieškom ^ ženklo pozicijos

n = str2num( maxXOrder( charPos+1:end)); % auksciausias x laipsnio rodiklis daugianaryje (daugianario eile)

% suformuojama visu x laipsniu eile:

for i = 1 : n + 1,

orders\_full(i) = x^(n - i + 1);

% tarkim n = 5; tada orders\_full = [x^5, x^4, x^3, x^2, x^1, 1]

end

% koeficientu eile papildoma nuliniais nariais:

for i = 1 : n + 1

j = find( orders == orders\_full(i));

if j > 0, % jei rado - įrašom reikšmę

Cf(i) = coeffs1(j);

CfNeg(i) = coeffs1\_neg(j);

else, % jei nerado - vietoj koeficientų įrašom nulius

Cf(i)=0;

CfNeg(i)=0;

end

end

% koeficientas prie auksciausio x laipsnio turi buti teigiamas:

Cf = Cf / Cf(1); % f(x) koeficientai

CfNeg = CfNeg / CfNeg(1); % f(-x) koeficientai

% paverčiam skaičiais

CfValues = eval(Cf);

CfNegValues = eval(CfNeg);

%----------------------------/Koef skaičiavimas----------------------------

% ------------- Grubus ivertis----------------------------------------:

R = max( abs( CfValues( 2 : end))) / CfValues(1) + 1; % taikoma grubaus ivercio formule % nuo 2:end todėl nes 1 <= i <= n-1

fprintf('Grubus įvertis [-%g; %g]\n', R, R)

% grafinis funkcijos, saknu ir grubaus ivercio intervalo pavaizdavimas:

points = 1000;

t = -R : R / points : R;

figure(1);

grid on;

hold on;

plot(t, fnk(CfValues, t), 'g-');

plot( [-R,R], [0 0], 'r\*');

% -------------/Grubus ivertis----------------------------------------:

% ------------ Tikslesnis ivertis-------------------------------------:

% teigiamoms saknims:

neg\_ind = find( CfValues( 2 : end) < 0); % suranda neigiamų indeksus

if ~isempty(neg\_ind)

B = max( abs( CfValues( neg\_ind + 1))); % skaičiuoja B; + 1, nes paieška neg\_ind nuo vektoriaus 2 nario

k = neg\_ind(1);

Rpos = 1 + ( B / CfValues(1)) ^ (1 / k);

else

Rpos = 0;

end

% neigiamoms saknims:

neg\_ind1 = find( CfNegValues( 2 : end) < 0);

if ~isempty(neg\_ind1)

B = max( abs( CfNegValues( neg\_ind1 + 1)));

k = neg\_ind1(1);

Rneg = 1 + ( B / CfNegValues(1)) ^ (1 / k);

else

Rneg = 0;

end

plot( [-Rneg, Rpos], [0 0], 'bp'); % pavaizduojami tikslesnio įverčio intervalo galai

% ------------/Tikslesnis ivertis-------------------------------------:

fprintf('Tikslesnis įvertis [-%g; %g]\n', Rneg, Rpos)

% ------------ Daugianario šaknys--------------------------------------

fRoots = roots(CfValues);

plot(fRoots, 0 \* fRoots, 'go');

fRootsString = mat2str(fRoots)

legend('kreive f(x)', 'grubus saknu intervalo ivertis', 'tikslesnis saknu intervalo ivertis', strcat('f(x)=0 saknys ', fRootsString));

% ------------/Daugianario šaknys--------------------------------------

end

function p = fnk(Cf, x)

% Apskaiciuoja daugianario reiksmes, kai argumentas yra x

% Kai x yra reiksmiu vektorius, p taip pat yra atitinkamu funkcijos reiksmiu vektorius

p = 0;

n = length(Cf) - 1;

for i = 1 : length(Cf),

p = p + Cf(i) \* x.^ (n - i + 1); % veiksmas < .^ > reiskia, kad laipsniu keliami visi vektoriaus x elementai

end

return

end

***Failas Niutono.m***

function Niutono

clc, close all, clear all

syms f x;

f = 64 \* x^4 - 576 \* x^3 + 10 \* x^2 + 9 \* x; % simboline israiska, argumentas x

%f=x.\*sin(x)

df=diff(f,x) %isvestine

char(f)

ff=inline(char(f),'x') %char(f) vercia traktuoti ne kaip simbol kintamuosius bet kaip eilute

dff=inline(char(df),'x')

xx = [-2:0.01:10];

figure(1), hold on, grid on,

plot(xx, subs(f,x,xx), 'b-')

x0=10

prec=1e-9

alpha=1

nIters = 1;

plot(x0, 0,'rp'), plot([x0,x0],[0 ff(x0)], 'g-') %piešiam tašką ir vertikalę

while 1

prec1=abs(ff(x0));

fprintf(1,'iter= %g tikslumas=%g\n', nIters, prec1);

if prec1 < prec,

break,

end;

%formule x0 - fjos reiksme taske x0 dalinta fjos isvedstines....

x=x0-alpha\*ff(x0)/dff(x0); %pasako kiek sekantis artinys nutoles

plot(x0, 0,'rp'), plot([x0,x],[ff(x0) 0], 'g-')

x0=x;

pause

nIters = nIters + 1;

end

fprintf(1, '\n\nsprendinys=%g', x0);

end

1. *Tiesinių lygčių sistemų sprendimo algoritmai*. **Duota** tiesinių lyčių sistema [A][X]=[B].
   1. Išspręskite šias sistemas, naudodami nurodytą metodą.
   2. Patikrinkite gautus sprendinius ir skaidas, įrašydami juos į pradinę lygčių sistemą.
   3. Patikrinkite gautus sprendinius, naudodami MATLAB lygčių sistemų sprendimo (visiems metodams) ir matricų skaidos funkcijas (skaidos metodams).

Tame pačiame *Word* faile pateikite **rezultatus**:

1. Rezultatų lentelę.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprendinys [X]** | **Skaidos rezultatai (skaidos metodams)** | **Daugikliai ir iteracijų skaičius (iteraciniams metodams)** | **Sprendinio patikrinimas** | | |
| **[A][X]-[B]** | **Lygčių sistemos sprendimo funkcija (nurodykite konkrečią)** | **Skaidos funkcija (nurodykite konkrečią)** |
| -17.8881  5.0651  2.0319  33.3596  -3.1267 |  | 2  2  2  2  2  Iteracijų 131 | 1.0e-007 \*  0.4777  0.1546  -0.1686  -0.5732  -0.0759 | linsolve(A0, b0) |  |

1. Programos kodą.

**Pastaba**. Programos turi atlikti **tik užduotyje nurodytus veiksmus** ir **tik nurodytais metodais**.

***Failas PILS.m***

function PILS

clc, close all, clear all

A = [ 5 1 2 4 1;

1 6 1 2 2;

2 1 15 1 1;

4 2 1 7 1;

1 2 1 1 5];

b = [50;

75;

30;

171;

12];

n = length(b);

A0 = A; % saugom pradines reikšmes, nes išdarkom pozicijas

b0 = b;

eps = 1e-10;

itmax = 1000;

alph = 2;

alpha = [alph; alph; alph; alph; alph]; % taisymo koeficientai

% bandom matricą A sutvarkyti

for i = 1 : n

% darom kad gautumėm vienetukus įstrižainėj

b(i) = b(i) / A(i, i); % eilutė numeris i ir visi stulpeliai bus keičiami į tą eilutę padalintą iš įstrižainės nario.

A(i, :) = A(i, :) / A(i, i); % eilutė numeris i ir visi stulpeliai bus keičiami į tą eilutę padalintą iš įstrižainės nario.

A(i, i) = A(i, i) - alpha(i); % įstrižainė - alpha

end

A

b

% pradinis artinys

x = zeros(n, 1); % n eilučių, 1 stulpelis

for i = 1 : itmax

x1 = b - A \* x; % matricų daugyba % x1 reikalingas nes skaičiuosim tikslumą pagal buvusį x.

x1 = x1 ./ alpha; % apversti nereikia, nes naudojam tik įstrižainę

tikslumas = norm(x1 - x) / ( norm(x1) + norm(x));

fprintf(1, 'iteracija %d, tikslumas %g\n', i, tikslumas);

if tikslumas < eps,

break,

end

x = x1;

end

disp('spendinys');

x1

disp('matlab linsolve sprendimas');

x = linsolve(A0, b0)

% tikslumo tikrinimas

liekana = A0 \* x1 - b0

disp('bendra santykine paklaida:'), disp( norm(liekana) / norm(x));

end