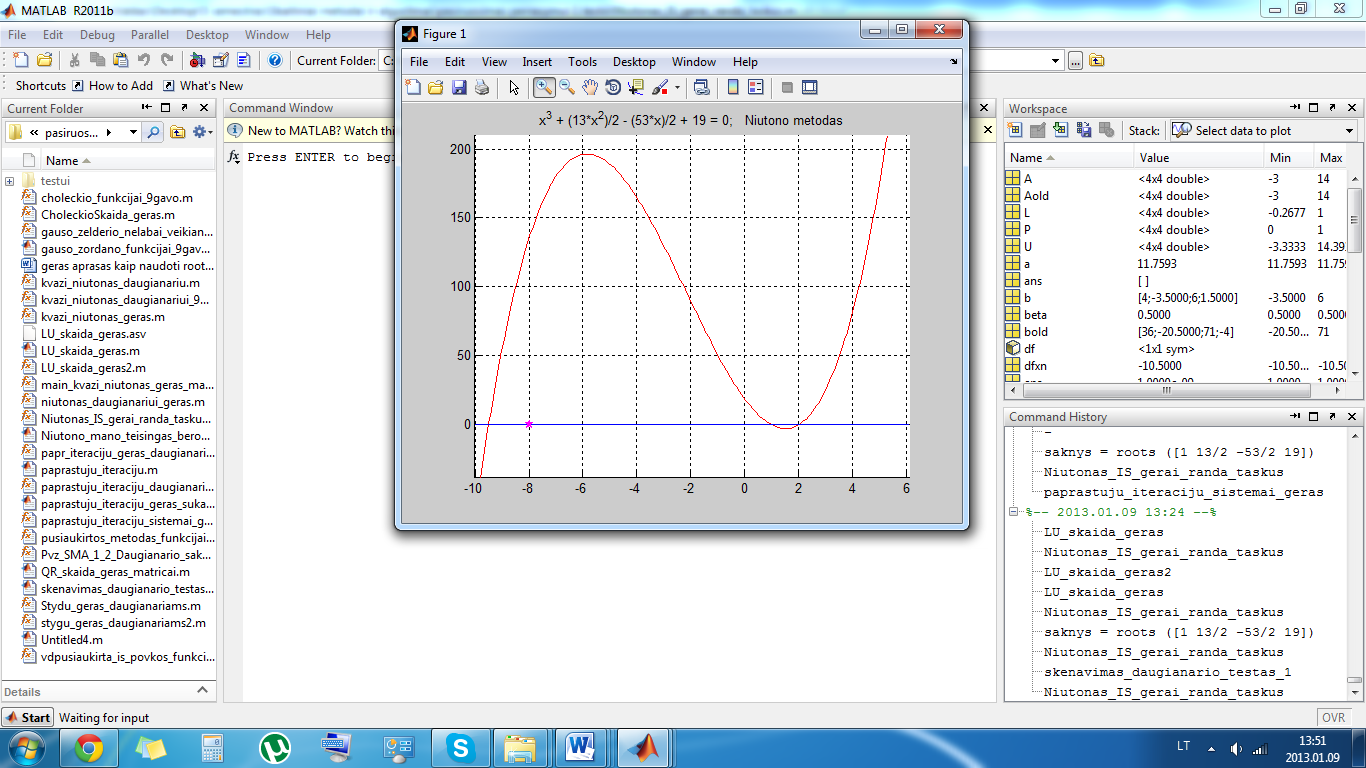
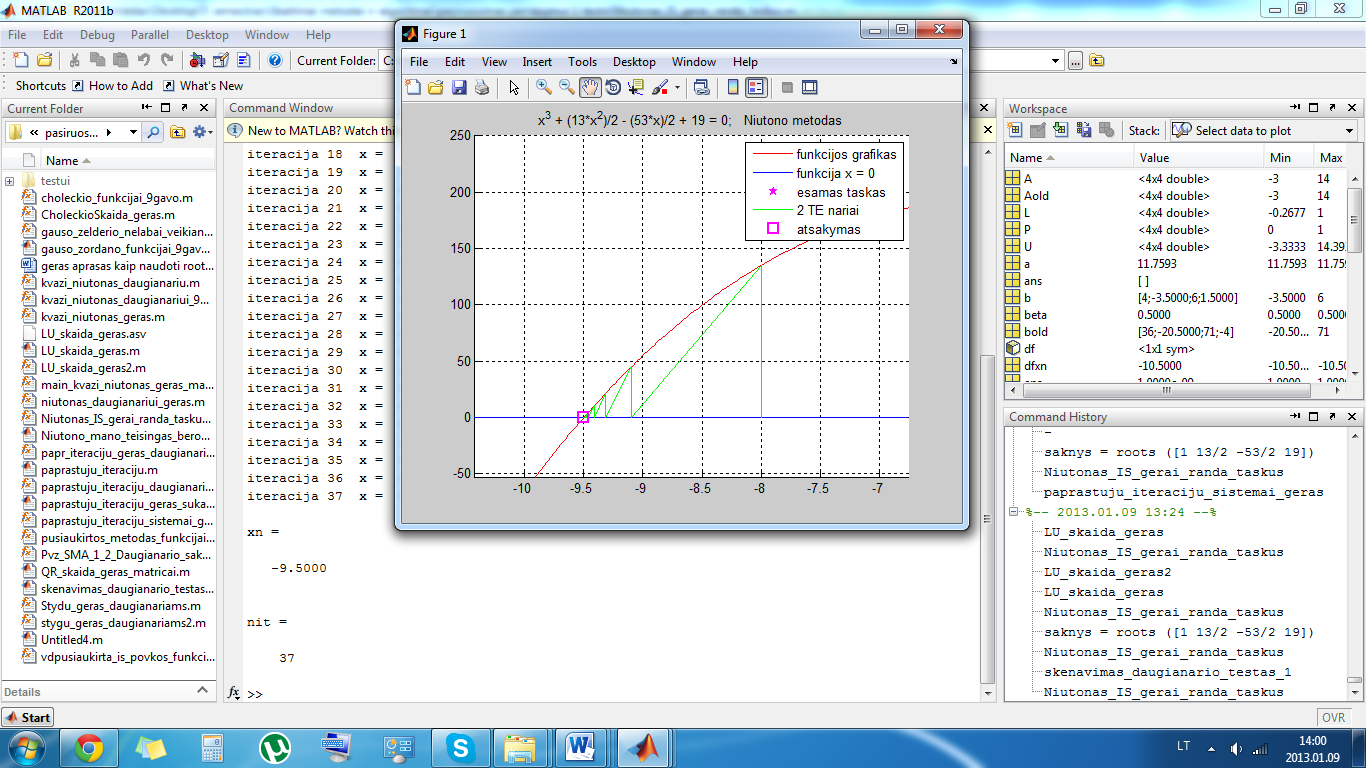
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Grupė** | **Pavardė Vardas** | **Savarankiško darbo Nr.** | **Lygčių Nr.** | **Sprendimo metodas** | |
| **daugianario** | **sistemos** |
| IFD-0 | Norkus Valdas | 18 |  | Niutono | LU skaidos |

**Niutono sprendimo rezultatai:**

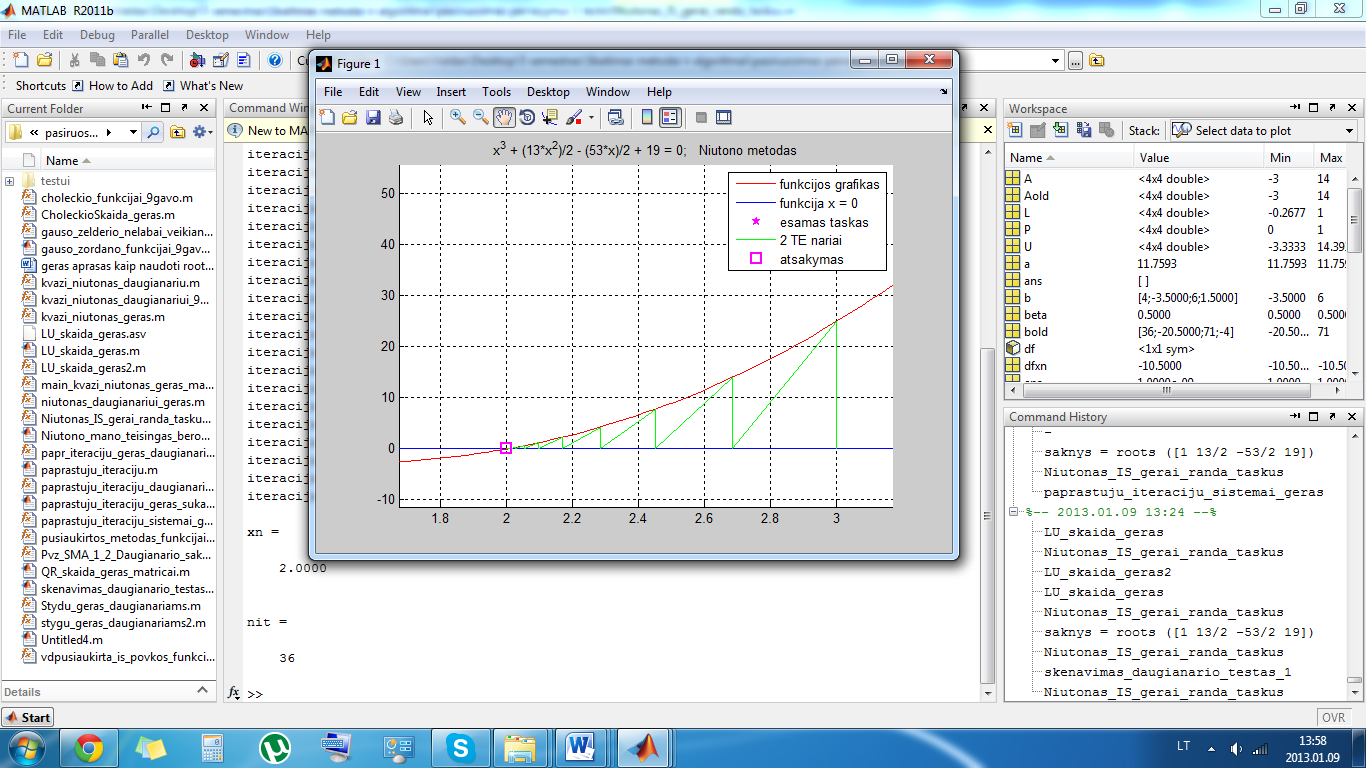
Funkcijos grafikas:



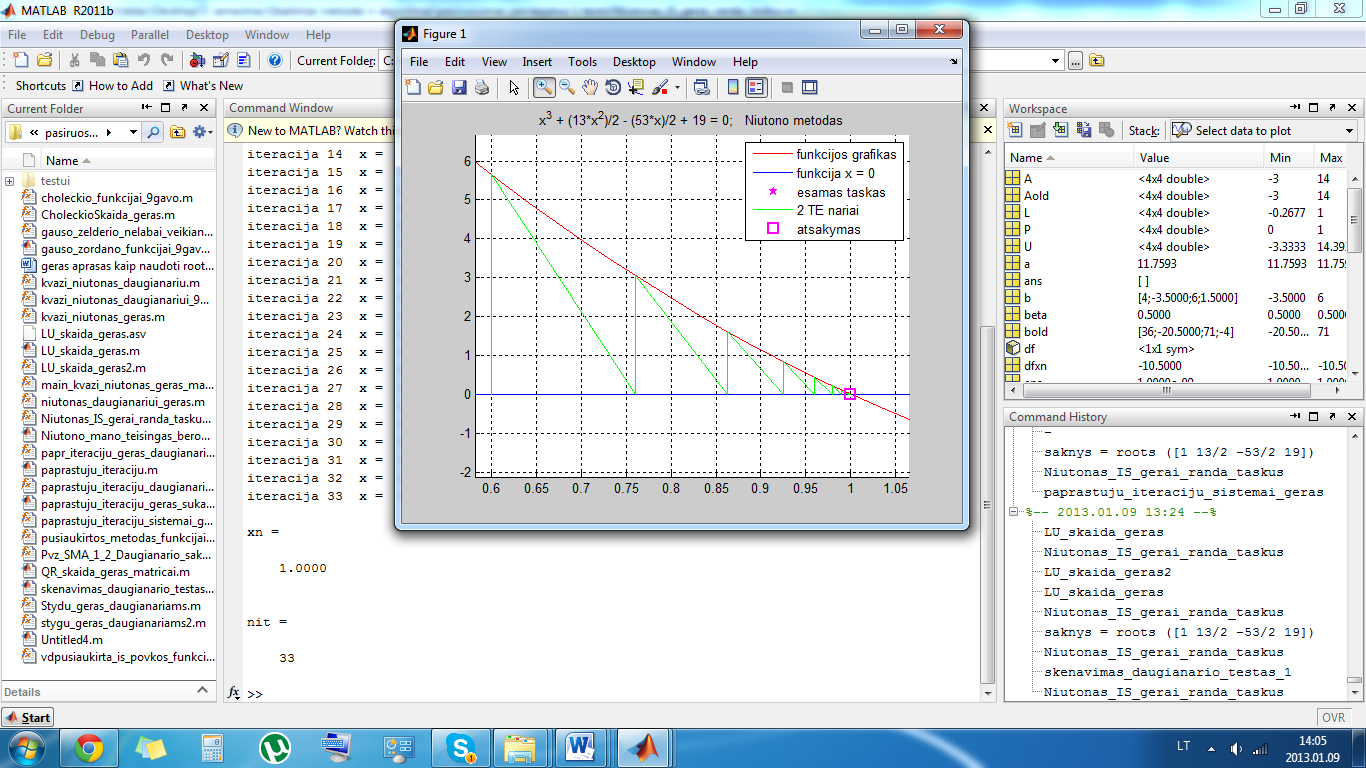
1 pav. Pavaizduotas funkcijos grafikas



2 pav. Rasta funkcijos pirma šaknis



3 pav. Rasta funkcijos antra šaknis



4 pav. Rasta funkcijos trečia šaknis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Metodas** | | | **Lygtis** | | **Šaknų intervalo įverčiai** | |
| Niutono | | |  | | Grubus 18  Tikslesnis [-16,575;18] | |
| **Šaknis** | **Intervalas**  ***arba* artinys** | **Tikslumas** | | **Iteracijų skaičius** | **Iteracijų pabaigos sąlygos** | ***roots* funkcijos rezultatas** |
| -9.5000 | -8 | 0.00000000060551963443 | | 37 |  | -9.5000 |
| 2 | 3 | 0.00000000062496208212 | | 36 |  | 2.0000 |
| 1 | 0.6 | 0.00000000085936058276 | | 33 |  | 1.0000 |

**Programos kodas:**

function Niutonas

clc, close all

syms f x

f = x^3+13/2\*x^2-53/2\*x+19;

x0 = 0.6;

alpha = 0.5;

range = [-15,15];

eps = 1e-9;

df = diff(f,x);

% braizomas funkcijos grafikas

npoints = 1000;

xrange = range(1) : (range(2)-range(1))/(npoints-1) : range(2);

figure(1); grid on; hold on;

str = [char(f),' = 0; Niutono metodas'];

title(str);

h1 = plot(xrange,eval(subs(f,x,sym(xrange))),'r-');

h2 = plot(range,[0 0],'b-');

h3 = plot(x0,0,'mp','MarkerFaceColor','m');

input('Press ENTER to begin'), figure(1);

% lygties sprendimas

xn = x0;

prec = 1;

nit = 0;

nitmax = 100;

legend([h1,h2,h3], 'funkcijos grafikas', 'f-jos x = 0 grafikas', 'esamas taskas');

while prec > eps

nit = nit + 1;

if nit > nitmax, fprintf('Virsytas leistinas iteraciju skaicius');break;end

fxn = eval(subs(f,x,sym(xn)));

dfxn = eval(subs(df,x,sym(xn)));

xn1 = xn - alpha \* fxn/dfxn; % daugikliu alpha apribojamas x prieaugis

h4 = plot([xn,xn,xn1],[0,fxn,0],'g-');

delete(h3);

h3 = plot(xn1,0,'mp','MarkerFaceColor','m');

xn = xn1;

pause(0.1)

fxn = eval(subs(f,x,sym(xn)));

prec = abs(fxn);

fprintf(1,'iteracija %d x = %.12f prec = %.20f \n',nit,xn,prec);

legend([h1,h2,h3,h4], 'funkcijos grafikas', 'funkcija x = 0', 'esamas taskas', '2 TE nariai');

% TE - Teiloro eilute

end

h5 = plot(xn,fxn,'ms','MarkerSize',10,'LineWidth',2);

legend([h1,h2,h3,h4,h5], 'funkcijos grafikas', 'funkcija x = 0', 'esamas taskas', '2 TE nariai', 'atsakymas');

delete(h3);

xn

nit

end

**LU skaidos rezultatai:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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)** |
| 4.0000  -3.5000  6.0000  1.5000 | U =  6.0000 0 1.0000 4.0000  0 9.0000 0.6667 -3.3333  0 0 11.7593 -1.2963  0 0 0 14.3937  L =  1.0000 0 0 0  0.3333 1.0000 0 0  0.1667 0.1111 1.0000 0  0 0.2222 -0.2677 1.0000 |  | 1.0e-014 \*  0  0  0  0.3553 | linsolve | LU |

**Programos kodas:**

clc

A=[6 0 1 4;

2 9 1 -2;

1 1 12 -1;

0 2 -3 14]

b=[36;-20.5;71;-4]

n=size(A,1)

P=[1:n]

Aold=A

bold=b

for i=1:n-1

[a,iii]=max(abs(A(i:n,i)));

A([i,i+iii-1],:)=A([i+iii-1,i],:);

P([i,i+iii-1])=P([i+iii-1,i]);

for j=i+1:n

r=A(j,i)/A(i,i);

A(j,i+1:n)=A(j,i+1:n)-A(i,i+1:n)\*r;

A(j,i)=r;

end

end

b=b(P)

% 1-as atvirkstinis etapas, sprendziama Ly=b, y->b

for i=2:n, b(i,:)=b(i,:)-A(i,1:i-1)\*b(1:i-1); end

% 2-as atvirkstinis etapas , sprendziama Ux=b, x->b

for i=n:-1:1, b(i)=(b(i)-A(i,i+1:n)\*b(i+1:n))/A(i,i); end

fprintf(1, 'Sprendinys:');

x=b

fprintf(1, 'AX-B:');

Aold\*x-bold

fprintf(1, 'Sprendinio patikrinimas:');

linsolve(Aold, bold)

%Gauname L ir U matricas

L=[];

U=[];

for i=1:n

U=[U; zeros(1, i-1), A(i,i:n)];

end

for i=1:n

L=[L; A(i,1:i-1) 1 zeros(1, n-i)];

end

fprintf(1, 'Gauta skaida');

U

L

fprintf(1, 'Atstatyta pradinė matrica');

A=L\*U;

A=A(P,:)

fprintf(1, 'Skaidos patikrinimas:');

[L,U,P]=lu(Aold);

U

L