Prvi primer

Neka je u M-fajlu $vektor.m$ zadat vektor X.

$vektor.m$

```
X = [1 4 2 -5 -2 9];
```

- Napisati M-fajl sa funkcijom $[p, y] = nule(x)$ koja formira polinom p kome su nule pozitivne vrednosti vektora X i računa vrednost tog polinoma u tački x.

```
function [p, y] = nule(x)

vektor;

% I način - for petlja
Y = [];
for i = 1:length(X)
    if X(i) > 0
        Y = [Y, X(i)];
    end
end

% II način - korišćenje ugrađene funkcije find()
% Y = X(find(X > 0));

% funkcija poly() vraća polinom čije su nule elementi vektora koji prosleđujemo kao
argument
p = poly(Y);
y = polyval(p, x);
```

- Napisati M-fajl sa funkcijom $i = izvod(x)$ koja za uneti argument x računa vrednost izvoda u tački x polinoma dobijenog pozivom funkcijskog fajla $nule.m$

```
function i = izvod(x)

% Svejedno je koju vrednost x ćemo proslediti kao argument pri pozivu funkcije
% [p, y]=nule(x) jer nam je bitan samo polinom koji se formira.

% Kada funkcijskom fajlu koji vraća više vrednosti kao rezultat dodelimo jednu promenljivu,
% dobićemo prvu vrednost rezultata. U našem slučaju, potreban nam je samo polinom, ne i
%vrednost polinoma u tački x=1
p = nule(1);

% k=polyder(p) vraća polinom k koji predstavlja prvi izvod polinoma p
% dok k=polyder(p,q) vraća polinom k koji predstavlja prvi izvod polinoma p*q.
i = polyval(polyder(p), x);
```

- Nacrtati grafik polinoma $x^4 + x^2 - 1$ na intervalu $[-2, 2]$ koristeci ekvidistantnu mrežu sa 50 tačaka

```
% polinom se predstavlja vektorom njegovih koeficijenata
% x^4 + x^2 -1 = 1*x^4 + 0*x^3 + 1*x^2 + 0*x^1 - 1*x^0
p = [1, 0, 1, 0, -1];

% ekvidistantna mreža od 50 tačaka na [-2, 2]
x = linspace(-2, 2, 50);

% vrednost polinoma u čvorovima mreže
y = polyval(p, x);

plot(x, y);

% izjednačavanje osa
axis equal

title('Grafik polinoma x^4+x^2-1');
xlabel('x-osa');
ylabel('y-osa');
```

Interpolacija i diferenciranje

1. zadatak

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 \ldots, x_n]$ i $F = [f_1, f_2 \ldots, f_n]$ (od kojih je prvi strogo rastući) za tu tablično zadatu funkciju. Tablica ne mora biti ekvidistantna.

$tablica.m$

```
% Komandni fajl sa vektorima X i F

% X=[0 0.2 0.4 0.6 0.8 1];
% Umesto da navodimo sve ove elemente, možemo pisati
% X je vektor vrednosti od 0 : sa korakom 0.2 : do 1
X = 0:0.2:1;

% F=[1 2 3 4 5 6];
F = 1:6;

% primer sa vežbi:
% X=[100 121 144];
% F=[10 11 12];
```

- Napisati M-fajl $novatablica.m$ u kom se prethodna tablica proširuje do nove dodavanjem čvorova $\frac{x_i+x_{i+1}}{2}, i = 1, \ldots n-1$ i računanjem vrednosti funkcije f u njima korišćenjem formule: $f(\frac{x_i+x_{i+1}}{2}) = \frac{f(x_i)+f(x_{i+1})}{2}, i = 1, \ldots n-1$.

```
% Komandani fajl koji od stare tablice u komandnom fajlu tablica.m (X i F)
% formira novu tablicu (X1 i F1)

tablica; % Učitavamo podatke iz tablica.m (X i F)
n = length(X); % Dužina niza X

% Poželjno je formirati nula nizove dužine 2*n-1 (dužina proširenog niza)
kako ne bismo u %svakom prolasku for petlje menjali dimenziju nizova
```

```
X1 = zeros(1, 2*n-1);
F1 = zeros(1, 2*n-1);

for i = 1:2:2*n-1
    % indeks mora biti ceo broj
    X1(i) = X((i + 1)/2);
    F1(i) = F((i + 1)/2);
    % ili    X1(i)=X(round(i/2));
    %        F1(i)=F(round(i/2));
end

% Računamo nove elemente
for i = 2:2:2*n-1
    X1(i)=(X1(i+1)+X1(i-1))/2;
    F1(i)=(F1(i+1)+F1(i-1))/2;
    % ili    X1(i) = (X(i/2) + X(i/2+1)) / 2;
    %        F1(i) = (F(i/2) + F(i/2+1)) / 2;
end

% Funkcije za zaokruživanje: round(), ceil() i floor()
% round(1.8)=2
% round(1.3)=1
% round(1.5)=2
% ceil(1.3)=2
% floor(1.8)=1
```

- Napisati M-fajl $Lagr1.m$ sa funkcijom $L = Lagr1(x)$ koja za uneti argument x vraća približnu vrednost funkcije f u toj tački izračunatu pomoću Lagranžovog interpolacionog polinoma L korišćenjem svih vrednosti iz nove tablice.

```
function L = Lagr1(x)

% Računamo vrednost Lagranžovog interpolacionog polinoma u tački x
% Argumenti funkcije:
%   x - (broj) tačka u kojoj računamo vrednost Lagranžovog int. polinoma
% Funkcija vraća:
%   L - vrednost Lagranžovog int. polinoma u tački x
% NAPOMENA: ne konstruišemo polinom, ne nalazimo koeficijente polinoma!

novatablica;
L = 0;
n = length(X1);
for i = 1 : n
    p = 1;
    for j = 1 : n
        if i ~= j
            p = p * (x - X1(j)) / (X1(i) - X1(j));
        end
    end
    L = L + p * F1(i);
end
```

2. zadatak

Neka je funkcija f zadata eksplicitno komandnim M-fajlom $funkcija.m$.

$funkcija.m$

```
% Primer funkcije kod koje se sa povećanjem broja čvorova povećava i greška
(Rungeov fenomen)
f = @(x) 1 ./ (1 + 25 * x.^2);

% Drugi način definisanja funkcije
% f=inline('1./(1+25*x.^2)');

% neki drugi primeri funkcija (anonimne funkcije)
%f=@(x) sin(x)/3+cos(x).^2;
%f=@(x) 45./(10.*x.^2+1);
```

- Napisati M-fajl $tablica.m$ sa funkcijom $[X, Y] = tablica(a, b, n)$ koja tabelira zadatu funkciju f na intervalu $[a, b]$ sa n čvorova.

```
function [X, Y] = tablica(a, b, n)

% Agrumenti funkcije:
%   a i b - granice intervala
%   n - broj tačaka na koji delimo interval [a,b]
% Funkcija vraća:
%   X - čvorove interpolacije
%   Y - vrednosti funkcije u tim čvorovima

% Poziv funkcije: [X Y] = tablica(-2,2,5)

funkcija;

% I način
% n ravnomerno raspoređenih tačaka od a do b => broj podintervala  je n-1
% dužina svakog od njih (korak h) iznosi
h = (b - a) / (n - 1);

for i = 1 : n
    X(i) = a + (i - 1) * h;
    Y(i) = f(X(i));
    %Y(i)=feval(f,X(i));
end

% II način - ugrađena funkcija linspace()
% X = linspace(a, b, n);
% Y = f(X);
```

- Napisati M-fajl $Lagr1b.m$ sa funkcijom $[L, y] = Lagr1b(x, a, b, n)$ koji formira i vraća koeficijente Lagranžovog interpolacionog polinoma L formiranog koristeći sve vrednosti iz tablice, kao i vrednost formiranog polinoma u tački x.

```
function [L, y] = Lagr1b(x, a, b, n)

% Poziv funkcije, npr: [L,y]=Lagr1b(0.5,-2,2,5)
```

```
% Argumenti funkcije:
%   x - tačka u kojoj se računa vrednost Lagranžovog int. polinoma
%   a i b - granice intervala [a,b]
%   n - broj čvorova
% Funkcija vraća:
%   L - koeficijente odgovarajućeg Lagranzovog int. polinoma
%   y - vrednost polinoma u tački x
%   NAPOMENA: sada konstruišemo polinom!

[X, Y] = tablica(a, b, n);
n = length(X);
L = zeros(1, n); % L je sada polinom

for i = 1 : n
    p = 1; % p je trenutno polinom nultog stepena, tj. p=1*x^0
    for j = 1 : n
        if i ~= j
            % Množimo 2 polinoma
            % [1 -X(j)] je polinom 1*x^1 - X(j)*x^0
            % NAPOMENA: paziti na razmak
            % [1 - X(j)] nije isto što i [1 -X(j)], prvi vektor je polinom
(1-X(j))*x^0
            p = conv(p, [1, -X(j)]/(X(i) - X(j)));
        end
    end
    L = L + p * Y(i);
end

% Računamo vrednost polinoma L u tački x
y = polyval(L, x);
```

- Uporediti grafike funkcije f i formiranog interpolacionog polinoma.

```
function grafik(a, b, n)

% Poziv funkcije:
% grafik(-1,1,5) interpolacija sa 5 tačaka
% grafik(-1,1,10) interpolacija sa 10 tačaka
% grafik(-1,1,11) interpolacija sa 11 tačaka

[L, y] = Lagr1b(1, a, b, n); %x je nebitan za grafik
funkcija;

% Delimo interval [a,b] na 100 tačaka
x = linspace(a, b);

% Želimo više grafika na jednoj slici (moramo uključiti hold on i kasnije
isključiti sa
% hold off). Svaki poziv plot() izmedju ove dve naredbe crtaće novi
% grafik i pritom se grafici prethodnog pozivanja funkcije plot() neće
obrisati

hold on

% Crtamo prvi grafik funkcije
plot(x, f(x));
% Crtamo drugi grafik polinoma
```

```
plot(x, polyval(L, x), 'r');

hold off

legend('funkcija f', 'interpolacioni polinom')
% Izjednačavamo ose
axis equal
```

3. zadatak

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 \ldots, x_n]$ i $Y = [y_1, y_2 \ldots, y_n]$ za tu tablično zadatu funkciju.

$tablica.m$

```
X = 15:5:55;
Y = [0.2588, 0.3420, 0.4226, 0.5000, 0.5736, 0.6428, 0.7071, 0.7660, 0.8192];
```

- Napisati M-fajl $tablicaCheck.m$ sa funkcijom $t = tablicaCheck()$ koja vrši proveru da li je tablica u komandnom fajlu $tablica.m$ ekvidistantna i da li je niz X zadat u strogo rastućem poretku. Ukoliko su oba uslova ispunjena funkcija vraća vrednost 1, u suprotnom vraća vrednost 0 i u oba slučaja ispisuje odgovarajuću poruku.

```
function t = tablicaCheck()

% Funkcija vraća:
%       t=1 - ukoliko je sve u redu
%       t=0 - ukoliko nije ispunjen jedan od ova dva uslova

tablica;
r = all(diff(X) > 0);
h = X(2) - X(1);
% ekv=all(diff(X)==h); Radi samo za celobrojne vrednosti koraka h!
% zbog netačne reprezentacije relane brojeve nikada ne treba porediti na
jednakost
% već smatramo da su jednaki ako je apsolutna vrednost njihove razlike manja
od nekog %dovoljno malog broja
ekv = all(abs(diff(X)-h) <= 1e-10);
t = r && ekv;
if t == 1
    disp('Tablica je ekvidistantna i vektor X je zadat u strogo rastucem
poretku.')
else
    disp('Nisu ispunjeni uslovi zadatka!')
end
```

- Napisati M-fajl $polozaj.m$ sa funkcijom $polozaj(x)$ koja za uneti argument x vraća vrednost 1 ukoliko je $x < x_2$, 2 ukoliko je $x > x_{n-1}$ i 0 inače.

```
% Određuje položaj tačke x u odnosu na čvorove interpolacije
function pol = polozaj(x)

tablica;
n = length(X);
```

```
if x < X(2)
    pol = 1;
else if x > X(n-1)
        pol = 2;
    else
        pol = 0;
    end
end
```

- Napisati M-fajl $Njutn.m$ sa funkcijom $Njutn(x)$ koja ukoliko su svi uslovi ispunjeni, vraća približnu vrednost funkcije f u tački x izračunatu korišćenjem I (II) Njutnovog interpolacionog polinoma, ako je vrednost funkcije $polozaj$ u tački x jednaka $1(2)$, odnosno izdaje odgovarajuću poruku ukoliko je $polozaj(x) = 0$.

```
function nj = Njutn(x)

if tablicaCheck() == 1 % da li su ispunjeni uslovi zadatka
    if polozaj(x) == 1 % da li je x na početku tablice
        nj = Njutn1(x);
    else
        if polozaj(x) == 2 % da li je x na kraju tablice
            nj = Njutn2(x);
        else
            error('Tacka se nalazi u sredini');
        end
    end
else
    error('Nisu ispunjeni uslovi zadatka');
end

  % Ugrađena funkcija error() prekida izvršavanje progrma i vraća poruku
(string koji dobija
  % kao argument) kojom nas obaveštava zašto je prekinutno izvršavanje.
```

```
function nj1 = Njutn1(x)

% Pomoćna funkcija koju koristi Njutn.m za računanje vrednosti I Njutnovog
% interpolacionog polinoma u tački x

tablica;
n = length(X);

% KONAČNE RAZLIKE:
k_razlike = zeros(n, n-1);

% For petlja koja popunjava prvu kolonu tabele konačnih razlika, jer se
% one računaju oduzimanjem odgovarajućih elemenata vektora Y
for i = 1 : n-1
    k_razlike(i, 1) = Y(i+1) - Y(i);
end

% For petlja koja računa konačne razlike od drugog do n-1 reda
% Svako izračunavanje se vrši oduzimanjem odgovarajućih vrednosti
% iz prethodne kolone tabele konačnih razlika
```

```
for j = 2 : n-1 % po kolonama (red konačne razlike)
    for i = 1 : n-j % po vrstama (poslednji čvor za koje se može naći k.r.
reda j je n-j)
        k_razlike(i, j) = k_razlike(i+1, j-1) - k_razlike(i, j-1);
    end
end

disp(k_razlike);
% Želimo da matricu k_razlike nadovežemo na čvorove i vrednosti funk. u
čvorovima i da
% dobijemo tablicu kao na vežbama
% disp([X' Y' k_razlike])

% I NJUTNOV INTERPOLACIONI POLINOM:
y = Y(1);
h = X(2) - X(1);
q = (x - X(1)) / h;
% čuvamo početnu vrednost za q jer nam treba za formiranje svakog sledećeg
sabirka
Q = q;

for j = 1 : n-1
    y = y + q * k_razlike(1, j) / factorial(j);
    q = q * (Q - j);
end

nj1 = y;
% Samo računamo vrednost polinoma u tački, ne konstruišemo polinom!
```

```
function nj2 = Njutn2(x)

% Pomoćna funkcija koju koristi Njutn.m za racunanje II Njutnovog
% interpolacionog polinoma u tački x

tablica;
n = length(X);

% KONAČNE RAZLIKE:
k_razlike = zeros(n, n-1);

for i = 1 : n-1
    k_razlike(i, 1) = Y(i+1) - Y(i);
end

for j = 2 : n-1
    for i = 1 : n-j
        k_razlike(i, j) = k_razlike(i+1, j-1) - k_razlike(i, j-1);
    end
end

disp(k_razlike);

% II NJUTNOV INTERPOLACIONI POLINOM:
y = Y(end);
h = X(2) - X(1);
q = (x - X(end)) / h;
Q = q;
```

```
for j = 1 : n-1
    y = y + q * k_razlike(n-j, j) / factorial(j);
    q = q * (Q + j);
end

nj2 = y;
```

4. zadatak

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 ..., x_n]$ i $Y = [y_1, y_2 ..., y_n]$ za tu tablično zadatu funkciju. Tablica ne mora biti ekvidistantna.

$tablica.m$

```
X = [2 2.5 3.5 4];
Y = sin(X);

% ili npr.
% Y=X+1
%Y=[3 3.5 4.5 5];
```

- Napisati M-fajl $tablicaCheck.m$ sa funkcijom $t = tablicaCheck()$ koja vrši proveru da li je niz X zadat u strogo rastućem poretku i da li je niz Y monoton. Ukoliko su oba uslova ispunjena funkcija vraća vrednost 1, u suprotnom vraća vrednost 0. Ukoliko neki od uslova nije ispunjen, funkcija ispisuje odgovarajuću poruku.

```
function t = tablicaCheck()

% Funkcija vraća:
%       t=1 - ukoliko je sve u redu
%       t=0 - ukoliko nije ispunjen jedan od ova dva uslova

tablica;
n = length(X);

provera_x = 1;
provera_y = 1;

% Da li je X strogo rastući?
for i = 1 : n-1
    if X(i) >= X(i+1)
        provera_x = 0;
        disp('Niz X nije rastuci');
        break
    end
end

% Da li je Y monoton?
i = 1;
if Y(2) > Y(1) % Pretenduje da bude strogo rastući
    % Operator && ne računa logičku vrednost drugog operanda, ako prvi ima
vr. 0
    while (i < n && Y(i+1) > Y(i)) % Bitan redosled, kako se ne bi
pristupalo Y(n+1)
```

```
        i = i + 1;
    end
else if Y(2) < Y(1) % Pretenduje da bude strogo opadajući
        while (i < n && Y(i+1) < Y(i))
            i = i + 1;
        end
    end
end

if i < n % prekinuto je izvršavanje jedne od while petlji jer nije ispunjen
uslov za Y
    provera_y = 0;
    disp('Niz Y nije monoton');
end

if provera_x == 1 && provera_y == 1
    % if (provera_x && provera_y)
    t = 1;
else
    t = 0;
end
% Ili umesto if-a:  t=proverax && proveray

% II Nacin:

  % provera_x=all(diff(X)>0);
  % provera_y=all(diff(Y)>0) || all(diff(Y)<0);
  % t=provera_x && provera_y;
```

- Napisati M-fajl $vredfunk.m$ sa funkcijom $y = vredfunk(x)$ koja za uneti argument x vraća

približnu vrednost funkcije f u toj tački izračunatu pomoću Njutnovog interpolacionog polinoma sa podeljenim razlikama korišćenjem svih vrednosti iz tablice.

```
function y = vredfunk(x)

tablica;
n = length(X);

% Da li su ispunjeni uslovi zadatka?
if tablicaCheck() == 0
    error('Nisu ispunjeni uslovi zadatka');
end

% Podeljene razlike
p_razlike = zeros(n, n-1);

% For petlja koja popunjava prvu kolonu tabele podeljenih razlika, jer se
% one racunaju oduzimanjem odgovarajucih elemenata vektora Y
for i = 1 : n-1
    p_razlike(i, 1) = (Y(i+1) - Y(i)) / (X(i+1) - X(i));
end

% For petlja koja računa podeljene razlike od drugog do n-1 reda
% Svako izračunavanje se vrši oduzimanjem odgovarajućih vrednosti
```

```
  % iz prethodne kolone tabele podeljenih razlika
  for j = 2 : n-1 % po kolonama (red podeljene razlike)
      for i = 1 : n-j % po vrstama
          p_razlike(i, j) = (p_razlike(i+1, j-1) - p_razlike(i, j-1)) / (X(i+j)
- X(i));
      end
  end

  disp('Stampamo tablicu po kolonama:');
  % "lepimo" vektore X i Y na početak tablice podeljenih razlika
  disp([X', Y', p_razlike]);

  % Njutnov interpolacioni polinom sa podeljenim razlikama
  y = Y(1);
  p = 1;
  for i = 1 : n-1
      p = p * (x - X(i));
      y = y + p * p_razlike(1, i);
  end

  disp(y);


  % II nacin za formiranje matrice podeljenih razlika - funkcija diff()

  % prazlike(1:n-1,1)=diff(Y)./diff(X);
  % for j=2:n-1
  %     prazlike(1:n-j,j)=diff(prazlike(1:n-j+1,j-1))./(X(1+j:n)-X(1:n-j))';
  % end
```

5. zadatak

Neka su u komandnom fajlu $podaci.m$ dati funkcija f i vektor X koji sadrži samo celobrojne vrednosti.

$podaci.m$

```
X = [-5 -4 -2 -1 2 3 5 6 7 8 9 10]
f = @(x) (x + 1) / 3;
```

- Napisati M-fajl $tablica.m$ sa funkcijom $[X1, Y1] = tablica()$ koja formira tablicu gde se vektor $X1$ sastoji samo od parnih vrednosti vektora X, a vektor $Y1$ su vrednosti eksplicitno zadate funkcije f u elementima vektora $X1$ zaokruženi na 3 decimale.

```
function [X1, Y1] = tablica()

podaci;

% Ugradjena funkcija find(uslov) vraća vektor pozicija onih elemenata u
vektoru (koji je sadržan u uslovu) koji ispunjavaju zadati uslov
% find(mod(X,2)==0) vratiće vektor pozicija (indeksa) parnih elemenata
vektora X
% X(find(mod(X,2)==0)) vraća parne elemente vektora X

X1 = X(find(mod(X, 2) == 0));
Y1 = f(X1);
Y1 = round(Y1.*1000) / 1000;
% Ugrađenoj funkciji round() kao drugi argument možete proslediti na koliko
decimala %želite da izvršite zaokruživanje
% Y1=round(Y1,3);
```

- Napisati M-fajl $inverz.m$ sa funkcijom $x = inverz(y)$ koja za zadatu vrednost y inverznom interpolacijom približno određuje x za koje je $f(x) = y$. (*Tablica neće biti ekvidistantna, pa koristimo Lagranžov interpolacioni polinom)

```
function x = inverz(y)

[X, Y] = tablica();
n = length(X);

% Pošto iz postavke zadatka tablica ne mora biti ekvidistantna, za inverznu
% interpolaciju najpre ćemo invertovati tablicu, a zatim odrediti vrednost
% Lagranžovog interpolacionog polinoma u tački y

inv_Tablica = zeros(2, n);
% U prvu vrstu upisujemo vektor Y
inv_Tablica(1, :) = Y;
% U drugu vrstu upisujemo vektor X
inv_Tablica(2, :) = X;

% Tablicu sortiramo po Y u rastućem poretku
% Ugradjena funkcija sortrows(M,br_kol) sortiraće matricu M po koloni br_kol
% Želimo sortiranje po prvoj vrsti matrice inv_Tablica, tj. po prvoj koloni
% transponovane matrice
inv_Tablica = (sortrows(inv_Tablica', 1))';

% Prikazujemo kako izgleda naša tablica nakon invertovanja i sortiranja
disp('Invertovana tablica:');
disp(inv_Tablica);

% Izvlačimo prvu vrstu invertovane tablice (čvorovi za interpolaciju inverza
funk. f) % u vektor Y1 i drugu vrstu (vrednosti inverza funkcije f) u vektor
X1
Y1 = inv_Tablica(1, :);
X1 = inv_Tablica(2, :);

% Računamo vrednost Lagranžovog interpolacionog polinoma  u tački y (ne
konstruišemo polinom!)
L = 0;
for i = 1 : n
```

```
    p = 1;
    for j = 1 : n
        if i ~= j
            p = p * (y - Y1(j)) / (Y1(i) - Y1(j)); % u vektoru Y1 se nalaze
čvorovi
        end
    end
    L = L + p * X1(i); % u vektoru X1 se nalaze vrednosti inverza funkcije f
end
```

6. zadatak

Neka je funkcija f zadata eksplicitno komadnim M-fajlom $funkcija.m$.

$funkcija.m$

```
f = inline('x.^2+x/2-exp(x)/4');
```

- Napisati M-fajl $tablica.m$ sa funkcijom $[X, Y] = tablica(a, b, n)$ koja formira ekvidistantnu tabelu funkcije f na segmentu $[a, b]$ sa n čvorova.

```
function [X, Y] = tablica(a, b, n)

% Poziv funkcije: [X Y] = tablica(1,4.5,10)

funkcija; % učitavanje komandnog fajla kako bismo imali na raspolaganju
funk. f

% Korak
h = (b - a) / (n - 1);

for i = 1 : n
    X(i) = a + (i - 1) * h;
    Y(i) = f(X(i));
    % računanje vrednosti funk. korišćenjem ugrađene funkcije feval()
    % Y(i) = feval(f, X(i));

end

% Ili u dva reda:

% X=linspace(a,b,n)
% Y=f(X);

% NAPOMENA: Funkciji f prosleđujemo vektor i želimo da kao rezultat dobijemo
vektor %vrednosti funkcije f u svim elementima vektora X, stoga je neophodno
je uvesti '.' %notaciju tj. '.^'  pri definisanju funkcije f (svaki element
iz X treba kvadrirati).
```

- Napisati M-fajl $promenaZnaka.m$ sa funkcijom $[c, d] = promenaZnaka(a, b, n)$ koja na osnovu nizova X i Y dobijenih pozivanjem funkcije $tablica(a, b, n)$ pronalazi i kao rezultat vraća prvi interval $[x_i, x_{i+1}]$ u kome funkcija menja znak $(c = x_i, d = x_{i+1})$. Pretpostavlja se da takav interval postoji.

```
function [c, d] = promenaZnaka(a, b, n)

% Poziv funkcije: [c,d] = promenaZnaka(1,5,10)

[X, Y] = tablica(a, b, n);
n = length(X);
for i = 1 : n-1
    if Y(i) * Y(i+1) < 0
        break
    end
end

% II način - korišćenjem ugrađene funk. find()
% koordinatno množimo vektore [y1 y2 ... yn-1] i [y2 y3 ... yn] i u vekoru
rezultata
% tražimo poziciju prvog negativnog elementa. Kada funk. find() kao drugi
% argument prosledite 1, ona će vratiti poziciju prvog elementa koji
ispunjava uslov

%i=find(Y(1:n-1).*Y(2:n)<0,1);

c = X(i);
d = X(i+1);
```

- Napisati M-fajl $nula.m$ sa funkcijom $nula(a, b, n)$ koja metodom inverzne interpolacije približno određuje nulu funkcije f na intervalu $[c, d]$, koristeći II Njutnov interpolacioni polinom zaključno sa konačnim razlikama reda 3. Kriterijum zaustavljanja iterativnog niza: $|q_i - q_{i-1}| \leq 10^{-4}, i = 2...$

```
function x = nula(a, b, n)

% Poziv funkcije: x=nula(1,4.5,10)
% Nakon tabeliranja na segmentu [c,d] nula će se nalaziti na kraju tablice

[c, d] = promenaZnaka(a, b, n);
% tabeliramo funk. na podintervalu [c,d] u dodatnih n tačaka
[X1, Y1] = tablica(c, d, n);

h = X1(2) - X1(1);
% x - nula funkcije

% TABLICA KONACNIH RAZLIKA
% X(1)    Y(1)    kR(1,1)     kR(1,2)     kR(1,3)
% X(2)    Y(2)    kR(2,1)     kR(2,2)     kR(2,3)
% X(3)    Y(3)    kR(3,1)     kR(3,2)     kR(3,3)
%  .        .        .            .            .
%  .        .        .            .            .
%  .        .        .            .            .
% X(n-3) Y(n-3) kR(n-3,1)   kR(n-3,2)   kR(n-3,3)
% X(n-2) Y(n-2) kR(n-2,1)   kR(n-2,2)
% X(n-1) Y(n-1) kR(n-1,1)
% X(n)    Y(n)

k_razlike = zeros(n, 3); % potrebne su nam konačne razlike do 3. reda

for i = 1 : n-1
```

```
    k_razlike(i, 1) = Y1(i+1) - Y1(i);
end

for j = 2 : 3
    for i = 1 : n-j
        k_razlike(i, j) = k_razlike(i+1, j-1) - k_razlike(i, j-1);
    end
end

% Štampamo tabelu konačnih razlika
disp([X1', Y1', k_razlike]);

% II NJUTNOV INTERPOLACIONI POLINOM
% y = Y(n)+q*k_razlike(n-1,1)+q*(q+1)*k_razlike(n-2,2)/2+...
%                             q*(q+1)*(q+2)*k_razlike(n-3,3)/6

% ODGOVARAJUAĆA ITERATIVNA FORMULA
% q =(y-Y(n)-q*(q+1)*k_razlike(n-2,2)/2-
%           -q*(q+1)*(q+2)*k_razlike(n-3,3)/6)/k_razlike(n-1,1)

y = 0; % tražimo nulu funkcije
% q0=0, q1=slobodan član prethodno napisanog polinoma po q
% u vektor q smeštamo elemente iterativnog niza
q = [0, (y-Y1(end)) / k_razlike(n-1, 1)];

while (abs(q(end)-q(end-1)) > 0.0001)
    q1 = q(end);
    q = [q, (y-Y1(end)-k_razlike(n-2, 2)*q1*(q1 + 1)/2- ...  % za prelazak u
novi red
        k_razlike(n-3, 3)*q1*(q1 + 1)*(q1 + 2)/6) / k_razlike(n-1, 1)];
end
disp('Nula funkcije je: ');
x = X1(end) + q(end) * h;
disp('iterativni niz q je: ');
q'
```

7. zadatak - domaći

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 \ldots, x_n]$ i $Y = [y_1, y_2 \ldots, y_n]$ (od kojih je prvi strogo rastući) za tu tablično zadatu funkciju. Tablica ne mora biti ekvidistantna.

$tablica.m$

```
X = [-2, -1.5, 0, 0.7, 0.9, 1.1];
Y = 2 * X - exp(X);
```

- Napisati M-fajl $izvod.m$ sa funkcijom $[X_i, Y_i] = izvod()$ u kom se na osnovu prethodne tablice formira tablica prvog izvoda funkcije f u tačkama $x_2, \ldots, x_{n-1}$ korišćenjem sledeće formule: $f'(x_i) = \frac{f(x_{i+1})-f(x_{i-1})}{x_{i+1}-x_{i-1}}$, gde je $Y_i = [f'(x_2), \ldots, f'(x_{n-1})]$, a $X_i = [x_2, \ldots, x_{n-1}]$.
- Napisati M-fajl $vredizvod.m$ sa funkcijom $y = vredizvod(x)$ koja za uneti argument x vraća približnu vrednost prvog izvoda funkcije f izračunatu korišćenjem Njutnovog interpolacionog polinoma sa podeljenim razlikama konstruisanog na osnovu svih vrednosti iz tablice formirane korišćenjem funkcije $izvod()$.

- Napisati M-fajl $nula.m$ sa funkcijom $nul = nula()$ koja metodom inverzne interpolacije približno određuje i vraća jednu nulu prvog izvoda funkcije f korišćenjem Njutnovog interpolacionog polinoma sa podeljenim razlikama (pretpostavka je da je prvi izvod monotona funkcija).

  test primer

```
>> [Xi,Yi] = izvod()

X =

   -1.5000         0    0.7000    0.9000


Yi =

    1.5677    1.1861    0.3782   -0.4760

>> y = vredizvod(0.4)
>>   Tablica podeljenih razlika:
   -1.5000    1.5677   -0.2544   -0.4090   -1.2727
         0    1.1861   -1.1541   -3.4635         0
    0.7000    0.3782   -4.2713         0         0
    0.9000   -0.4760         0         0         0

y =

    1.0637

>> nul = nula()
>>   Tablica podeljenih razlika:
   -0.4760    0.9000   -0.2341   -0.3805   -1.0745
    0.3782    0.7000   -0.8665   -2.5764         0
    1.1861         0   -3.9310         0         0
    1.5677   -1.5000         0         0         0

nul =

    0.6276
```

8. zadatak - domaći

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 \ldots, x_n]$ i $Y = [y_1, y_2 \ldots, y_n]$ (od kojih je prvi strogo rastući) za tu tablično zadatu funkciju. Tablica mora biti ekvidistantna (sa korakom h).

$tablica.m$

```
X = linspace(1,2,6);
f = @(x) exp(x) - 15 * x;
Y = f(X);
```

- Napisati M-fajl $drugiizvod.m$ sa funkcijom $[X2_i, Y2_i] = drugiizvod()$ u kom se na osnovu prethodne tablice formira tablica drugog izvoda funkcije f u tačkama $x_2, \ldots, x_{n-1}$ korišćenjem sledeće formule: $f''(x_i) = \frac{f(x_{i-1})-2f(x_i)+f(x_{i+1})}{h^2}$, gde je $Y2_i = [f''(x_2), \ldots, f''(x_{n-1})]$, a $X2_i = [x_2, \ldots, x_{n-1}]$.
- Napisati M-fajl $vred2izvod.m$ sa funkcijom $y = vred2izvod(x)$ koja za uneti argument x vraća približnu vrednost drugog izvoda funkcije f izračunatu korišćenjem I Njutnovog interpolacionog polinoma konstruisanog na osnovu svih vrednosti iz tablice formirane korišćenjem funkcije $drugiizvod()$.
- Napisati M-fajl $nula.m$ sa funkcijom $y = nula()$ koja metodom inverzne interpolacije približno određuje nulu drugog izvoda funkcije f (pretpostavka je da je drugi izvod monotona funkcija) koristeći I Njutnov interpolacioni polinom zaključno sa konačnim razlikama reda 3. Kriterijum zaustavljanja iterativnog niza: $|q_i - q_{i-1}| \leq 10^{-4}, i = 2 \ldots$

  test primer

```
>> [X Y Y2] = drugiizvod()

X =

    1.2000    1.4000    1.6000    1.8000

Y2 =

    3.3312    4.0687    4.9696    6.0698

>> y = vred2izvod(1.66)
>>   Tablica konacnih razlika:
    1.2000    3.3312    0.7375    0.1633    0.0362
    1.4000    4.0687    0.9008    0.1994         0
    1.6000    4.9696    1.1003         0         0
    1.8000    6.0698         0         0         0

y =

    5.2771

>> y = nula()
>>   Tablica konacnih razlika:
    1.2000    3.3312    0.7375    0.1633    0.0362
    1.4000    4.0687    0.9008    0.1994         0
    1.6000    4.9696    1.1003         0         0
    1.8000    6.0698         0         0         0

y =

   -0.1068
```

9. zadatak - domaći

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 \ldots, x_n]$ i $Y = [y_1, y_2 \ldots, y_n]$ (od kojih je prvi strogo rastući) za tu tablično zadatu funkciju. Tablica mora biti ekvidistantna (sa korakom h).

$tablica.m$

```
X = linspace(2, 4, 10);
Y = exp(X);
```

- Napisati M-fajl $izvod1.m$ sa funkcijom $izvod1(x)$ koja računa vrednost prvog izvoda tabelirane funkcije u tački x koristeći diferenciranje I Njutnovog interpolacionog polinoma zaključno sa konačnim razlikama reda 4.

```
    function izvod1(x)
    % Poziv funkcije: izvod1(2.1)

    tablica;

    n = length(X);
    % TABLICA KONAČNIH RAZLIKA

    k_razlike = zeros(n, 4);
    for i = 1 : n-1
        k_razlike(i, 1) = Y(i+1) - Y(i);
    end
    for j = 2 : 4
        for i = 1 : n-j
            k_razlike(i, j) = k_razlike(i+1, j-1) - k_razlike(i, j-1);
        end
    end

    disp([X', Y', k_razlike]);

    % Izvod I Njutnovog interpolacionog polinoma
    % y' = (k_razlike(1,1) + (q-0.5)*k_razlike(1,2) +
    % (3*q^2-6*q+2)*k_razlike(1,3)/6+(4*q^3-18*q^2+22*q-
6)/24*krazlike(1,4))/h

    h = X(2) - X(1);
    Q = (x - X(1)) / h;
    yi = (k_razlike(1, 1) + (Q - 0.5) * k_razlike(1, 2) + (3 * Q^2 - 6 * Q +
2) * k_razlike(1, 3) / 6 + (4 * Q^3 - 18 * Q^2 + 22 * Q - 6) / 24 *
k_razlike(1, 4)) / h;

    disp('Vrednost prvog izvoda funkcije f u tacki ');
    x
    disp('je: ');
    yi
```

- Napisati M-fajl $izvod2.m$ sa funkcijom $izvod2(x)$ koja računa vrednost drugog izvoda tabelirane funkcije u tački x koristeći diferenciranje I Njutnovog interpolacionog polinoma zaključno sa konačnim razlikama reda 4.

  test primer

```
  >>izvod2(2.3)

  Tablica konacnih razlika:
      2.0000    7.3891    1.8388    0.4576    0.1139    0.0283
      2.2222    9.2278    2.2963    0.5714    0.1422    0.0354
      2.4444   11.5241    2.8678    0.7136    0.1776    0.0442
      2.6667   14.3919    3.5814    0.8912    0.2218    0.0552
```

```
    2.8889   17.9733    4.4726    1.1130    0.2770    0.0689
    3.1111   22.4460    5.5857    1.3900    0.3459    0.0861
    3.3333   28.0316    6.9756    1.7359    0.4320         0
    3.5556   35.0073    8.7115    2.1679         0         0
    3.7778   43.7188   10.8794         0         0         0
    4.0000   54.5982         0         0         0         0

Vrednost drugog izvoda funkcije f u tacki

x =

    2.3000

je:

yi2 =

   9.9598
```

Integracija

10. zadatak

Neka je funkcija f zadata eksplicitno komadnim M-fajlom $funkcija.m$.

$funkcija.m$

```
f = inline('x.^2+1');
%f=@(x) x.^2+1;
```

- Napisati M-fajl $trapez.m$ sa funkcijom $I = trapez(a, b)$ koja približno računa i vraća vrednost određenog integrala funkcije f (granice integracije su a i b) korišćenjem uopštene Trapezne kvadraturne formule sa $n = 9$ čvorova.

```
function I = trapez(a, b)

% Korišćenjem Trapezne kvadraturne formule računamo približne vrednosti
integrala
% od a do b funkcije f
% Poziv funkcije: I = trapez(1, 5)

funkcija;
n = 9;
% čvorovi kvadraturne formule
X = linspace(a, b, n);
% korak
h = (b - a) / (n - 1); % ili h = X(2) - X(1)

Y = f(X);
%vrednost integrala
I = (h / 2) * (Y(1) + Y(n) + 2 * sum(Y(2:n-1)));
```

- Napisati M-fajl $simps.m$ sa funkcijom $I = simps(a, b)$ koja približno računa i vraća vrednost određenog integrala funkcije f (granice integracije su a i b) korišćenjem uopštene Simpsonove kvadraturne formule sa $n = 9$ čvorova.

```
function I = simps(a, b)

% Korišćenjem Simpsonove kvadraturne formule računamo vrednosti integrala
% od a do b funkcije f
% Poziv funkcije: I = simps(1, 5)
```

```
funkcija;
n = 9;

% čvorovi kvadraturne formule
X = linspace(a, b, n);

% korak
h = (b - a) / (n - 1); % ili h = X(2) - X(1)
Y = f(X);

% vrednost integrala
I = (h / 3) * (Y(1) + Y(n) + 2 * sum(Y(3:2:n-1)) + 4 * sum(Y(2:2:n-1)));
% U Simpsonovoj formuli čvorove sa neparnim indeksom množimo sa 4 (formula
sa vežbi)
% Pošto indeksiranje vektora u MATLAB-u kreće od 1 (a ne od 0), ovde ćemo sa
4 množiti
% elemente vektora sa parnim indeksima.
```

- Napisati M-fajl $vredfunk.m$ sa funkcijom $[X, Y] = vredfunk(k, p)$ koja približno izračunava vrednost funkcije $I(x) = \int_1^x f(t)dt$, kada se x kreće od 2 do $k \in \mathbb{N}, k \geq 2$ sa korakom 1. Ukoliko je $p = 1$ integrale računati koristeći uopštenu Simpsonovu kvadraturnu formulu (sa $n = 9$ čvorova), a u slučaju kada je $p = 2$ uopštenu Trapeznu kvadraturnu formulu (sa $n = 9$ čvorova). Funkcija vraća dva vektora: X sa vrednostima x_i i Y sa izračunatim vrednostima funkcije $I(x)$ u tačkama x_i.

```
function [X, I] = vredfunk(k, p)

% Iz postavke zadatka, vektor X uzima vrednosti od 2 do k, sa korakom 1, tj.
% X = 2:1:k (a možemo ga formirati i u okviru for petlje kako je urađeno u
nastavku)
% Vektor I sadrži vrednosti integrala od 1 do X(i) funkcije f
% Pozivi funkcija: [X, I]=vredfunk(5,1), [X, I]=vredfunk(5,2)

for i = 1 : k-1
    % X=[2 3 4 5 .... k]
    X(i) = i + 1;
    if p == 1
        % Svaki element vektora I predstavlja približnu vrednost integrala
dobijenu
        % Simpsonovom kvadraturnom formulom
        I(i) = simps(1, X(i));
    else
        % Svaki element vektora I predstavlja približnu vrednost integrala
dobijenu
        % Trapeznom kvadraturnom formulom
        I(i) = trapez(1, X(i));
    end
end
```

11. zadatak

Neka je funkcija f zadata eksplicitno komadnim M-fajlom $funkcija.m$.

$funkcija.m$

```
f = @(x) x.^2 + 1 + sin(x);
%f=inline('x.^2+1+sin(x)');
```

- Napisati M-fajl $integralt.m$ sa funkcijom $[I, briter] = integralt(a, b, tol)$ koja sa tačnošću tol računa i vraća približnu vrednost određenog integrala funkcije f (granice integracije su a i b) korišćenjem uopštene Trapezne kvadraturne formule. Funkcija vraća vrednost integrala i broj iteracija.

```
function [I, briter] = integralt(a, b, tol)

% Poziv funkcije: [I,briter] = integralt(0,2,1e-4)

n1 = 3; % Krećemo od 3 cvora, 2 podintervala
I1 = trapez(a, b, n1);

% Potrebna nam je vrednost kvadraturne formule sa duplo manjim korakom, a to
ćemo
% dobiti dodavanjem središnje tačke unutar svakog podintervala. Broj
podintervala
% je n1-1 što je ujedno i broj novododatih tačaka, pa je ukupan broj tačaka
2*n1-1
n2 = 2 * n1 - 1;
I2 = trapez(a, b, n2);

runge = abs(I2-I1) / 3; % Rungeova ocena greške Trapezne kv. formule
briter = 2; % Već su izračunate 2 približne vr. integrala I1 i I2

% Dokle god je Rungeova ocena greške veća od dozvoljene tačnosti, polovimo
% korak i računamo novu vrednost integrala
while runge > tol
    I1 = I2;
    n2 = 2 * n2 - 1;
    I2 = trapez(a, b, n2);
    runge = abs(I2-I1) / 3;
    briter = briter + 1;
end
I = I2;
```

```
function I = trapez(a, b, n)

% Pomoćni fajl za integralt.m
% Uopštenom Trapeznom kv. formulom računa približnu vrednost integrala
% funkcije f na segmetu [a,b] koristeći n čvorova.
% Kako ćemo u svakoj iteraciji poloviti korak, tj. povećavati
% broj čvorova, potrebno je broj čvorova n prosleđivati kao argument

funkcija;
X = linspace(a, b, n);
h = (b - a) / (n - 1); % korak (n - cvorova => n-1 intervala izmedju njih)
Y = f(X);

% Trapezna kvadraturna formula
I = (h / 2) * (Y(1) + Y(end) + 2 * sum(Y(2:end-1)));
```

- Napisati M-fajl $integrals.m$ sa funkcijom $[I, briter] = integrals(a, b, tol)$ koja sa tačnošću tol računa i vraća približnu vrednost određenog integrala funkcije f (granice integracije su a i b) korišćenjem uopštene Simpsonove kvadraturne formule. Funkcija vraća vrednost integrala i broj iteracija.

```
  function [I, briter] = integrals(a, b, tol)

% Poziv funkcije: [I,briter] = integrals(0,2,1e-4)

% NAPOMENA: Ovde je neophodno krenuti sa neparnim brojem čvorova!!!
  n1 = 3; % Krećemo od 3 cvora, 2 intervala
  I1 = simps(a, b, n1);
  n2 = 2 * n1 - 1;
  I2 = simps(a, b, n2);

  runge = abs(I2-I1) / 15; % Rungeova ocena greške Simpsonove kv. formule
  briter = 2;

  while runge > tol
      I1 = I2;
      n2 = 2 * n2 - 1;
      I2 = simps(a, b, n2);
      runge = abs(I2-I1) / 15;
      briter = briter + 1;
  end
  I = I2;
```

```
  function I = simps(a, b, n)

  % Pomoćni fajl za integrals.m
  % Uopštenom Simpsonovom kv. formulom računa približnu vrednost integrala
  % funkcije f na segmetu [a,b] koristeći n čvorova.
  % Kako ćemo u svakoj iteraciji poloviti korak, tj. povećavati
  % broj čvorova, potrebno je broj čvorova n prosleđivati kao argument

  funkcija;
  X = linspace(a, b, n);
  h = (b - a) / (n - 1);
  Y = f(X);

  % Simpsonova kvadraturna formula
  I = (h / 3) * (Y(1) + Y(end) + 2 * sum(Y(3:2:end-1)) + 4 * sum(Y(2:2:end-1)));
```

- Napisati M-fajl $grafik.m$ sa funkcijom $grafik(a, b)$ koja prikazuje grafik zavisnosti brzine konvergencije Simpsonove kvadraturne formule (plavo) i Trapezne kvadraturne formule (crveno), za različite tolerancije ($tol = 10^{-1}, \ldots, 10^{-6}$).

```
function grafik(a, b)

% Poziv funkcije: grafik(0,10)

tol = [1e-1 1e-2 1e-3 1e-4 1e-5 1e-6];
%tol=1./10.^(1:6);
```

```
briter_t = zeros(1, 6);
briter_s = zeros(1, 6);


for i = 1 : 6
    [~, briter] = integralt(a, b, tol(i)); % Vrednost I nam ne treba, zato ~
    briter_t(i) = briter;
    [~, briter] = integrals(a, b, tol(i));
    briter_s(i) = briter;
end

plot(tol, briter_t, 'r', tol, briter_s, 'b');
legend('Trapezna', 'Simpsonova')
xlabel('tolerancije')
ylabel('broj iteracija')
```

12. zadatak

Neka je funkcija f (koja ne mora biti samo pozitivna) zadata eksplicitno komandnim M-fajlom $funkcija.m$.

```
% Komandni fajl
%f = inline('1./(x+cos(x))');
f = inline('sin(x)');

% Funkciju možemo zadati i u okviru funkcijskog fajla
%function y = funkcija(x)
% y = sin(x);
```

- Napisati M-fajl $Runge.m$ sa funkcijom $r = Runge(S1, S2)$ koja vraća vrednost Rungeove ocene greške uopštene Simpsonove kvadraturne formule, ako su $S1$ i $S2$ njene vrednosti od kojih je jedna izračunata sa dvostruko manjim korakom u odnosu na drugu.

```
function r = Runge(S1, S2)

% Računamo Rungeovu ocenu greške Simpsonove kvadraturne formule
% S1 - vrednost izračunata Simpsonovom kvadraturnom formulom sa korakom H
% S2 - vrednost izračunata Simpsonovom kvadraturnom formulom sa korakom
h=H/2

k = 4;
r = abs(S1-S2) / (2^k - 1);
```

- Napisati M-fajl $zapremina.m$ sa funkcijom $V = zapremina(a, b, tol)$ koja koristeći uopštenu Simp- sonovu kvadraturnu formulu vraća zapreminu tela nastalog obrtanjem figure ograničene pravama $y = 0, x = a, x = b$ i funkcijom f oko ose Ox izračunatu sa tačnošću tol. (Za ocenu tačnosti koristiti funkciju $Runge$.)

```
function V = zapremina(a, b, tol)

% Poziv funkcije: V = zapremina(1, 3, 1e-4)
% Provera za f(x)=sin(x): V=pi*integral(g,1,3) gde je
```

```
% g = @(x) f(x).^2 (ugrađenoj funkciji integral kao prvi argument
% treba proslediti anonimnu funkciju)

n = 3;
h = (b - a) / (n - 1);

S1 = Simpson_zap(a, b, h);
H = h / 2; %polovimo korak
S2 = Simpson_zap(a, b, H);

% Dokle god je Rungeova ocena greške veća od dozvoljene tačnosti, polovimo
% korak i računamo novu vrednost integrala
while Runge(S1, S2) > tol
    S1 = S2;
    H = H / 2;
    S2 = Simpson_zap(a, b, H);
end

V = pi * S2;
```

```
function I = Simpson_zap(a, b, h)

% Pomoćni fajl za zapremina.m
% Računamo integral od a do b funkcije f(x)^2

funkcija;
% Potrebni su nam čvorovi da bismo dobili vektor vrednosti
% kvadrata funkcije f u zadatim čvorovima. Nađimo najpre broj podintervala
% dužine h na segmentu [a,b]
% n - broj podintervala.
n = (b - a) / h;

% Za n podintervala, imaćemo n+1 čvor (n+1 mora biti neparan)
X = linspace(a, b, n+1);

Y = f(X).^2;
I = (h / 3) * (Y(1) + 2 * sum(Y(3:2:end-1)) + 4 * sum(Y(2:2:end-1)) +
Y(end));
```

13. zadatak - domaći

- Formirati M-fajl $sistem.m$ sa funkcijom $[B, b] = sistem(d, t, n)$ koja formira sistem linearnih jednačina koji se dobija prilikom nalaženja koeficijenata kvadraturne formule oblika $\int_0^d t(x) f(x) dx \approx \sum_{i=0}^n A_i f(\frac{i*d}{n})$ koja treba da je tačna za polinome što je moguće većeg stepena. Funkcija treba da vraća matricu sistema i vektor desne strane.
- Formirati M-fajl $koeficijenti.m$ sa funkcijom $koeficijenti(d, t, n)$ koja određuje koeficijente A_i gore napisane kvadraturne formule. Dozvoljeno je korišćenje operatora \ za rešavanje sistema. (* Nakon sistema linearnih jednačina, zadatak se može rešavati i nekom od metoda za sisteme linearnih jednačina: LU, iterativna,...).

  test primer:

```
>> t=@(x) sin(x);

>> [B,b]=sistem(1,t,5)
```

```
B =
 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
 0 0.2000 0.4000 0.6000 0.8000 1.0000
 0 0.0400 0.1600 0.3600 0.6400 1.0000
 0 0.0080 0.0640 0.2160 0.5120 1.0000
 0 0.0016 0.0256 0.1296 0.4096 1.0000
 0 0.0003 0.0102 0.0778 0.3277 1.0000
b =
 0.4597
 0.3012
 0.2232
 0.1771
 0.1467
 0.1251
>> koeficijenti(1,t,5)
ans =
 0.0051
 0.0270
 0.1151
 0.0531
 0.2077
 0.0517
```

14. zadatak

- Napisati M-fajl $legendre\_poly.m$ sa funkcijom $L = legendre\_poly(n)$ koja formira i vraća niz SVIH Ležandrovih polinoma do stepena n na intervalu [-1, 1]. Nacrtati grafik svih formiranih Ležandrovih polinoma.

```
function L = legendre_poly(n)
% Funkcija formira i vraća niz svih Ležandrovih polinoma do stepena n

% Funkcija vraća cell array L, tako da je L{i}=Li-1
% tj. na i-toj poziciji nalazi se Ležandrov polinom stepena i-1

L=cell(1,n+1);
L{1} = 1; % L0 = 1
L{2} = [1, 0]; % L1 = x

% Ostali polinomi se računaju na osnovu rekurentne formule
% Li(x)=((2i-1)*x*Li-1(x) - (i-1)*Li-2(x))/i

% Množenje polinoma sa x ekvivalentno je dodavanju nule na kraj
% Dodavanje nule na početak ne menja vektor i vrši se kako bi se omogućilo
% sabiranje vektora, tj. kako bismo izjednačili njihove dimenzije
for i = 2:n
    L{i+1} = ((2 * i - 1) * [L{i}, 0] - (i - 1)* [0, 0, L{i-1}])/i;
end
celldisp(L);

% Crtamo grafik sa svim polinomima u različitim bojama
x = linspace(-1, 1);
colors = 'bgrcmyk'; % 7 različitih boja
axis equal

hold on
```

```
for i = 1:length(L)
    plot(X, polyval(L{i}, X), colors(mod(i, 7)+1));
end
hold off
```

- Napisati M-fajl $Cebisev\_poly.m$ sa funkcijom $C = Cebisev\_poly(n)$ koja formira i vraća niz SVIH Čebiševljevih polinoma do stepena n na intervalu [-1, 1]. Nacrtati grafik svih formiranih Čebiševljevih polinoma.

```
function C = Cebisev_poly(n)
% Funkcija formira i vraća niz svih Čebiševljevih polinoma do stepena n

% Funkcija vraća cell array C, tako da je C{i}=Ci-1
% tj. na i-toj poziciji nalazi se Čebiševljev polinom stepena i-1

C=cell(1,n+1);
C{1} = 1; %C0=1
C{2} = [1, 0]; %C1=x

% Ostali polinomi se računaju na osnovu rekurentne formule
%  Ci(x) = 2*x*Ci-1(x) - Ci-2(x)
for i = 2:n
    C{i+1} = 2 * [C{i}, 0] - [0, 0, C{i-1}];
end
celldisp(C);

%Crtmo grafik
X = linspace(-1, 1);
colors = 'bgrcmyk';

hold on
for i = 1:length(C)
    plot(X, polyval(C{i}, X), colors(mod(i, 7)+1));
end
hold off
```

- Napisati M-fajl $integrali.m$ sa funkcijom $integrali(f)$ koja korišćenjem ugrađene MATLAB funkcije integral() računa i štampa vrednosti sledećih integrala: $\int_{-1}^{1} f(x)dx$, $\int_{-1}^{1} f(x) \cdot sin(x)dx$, $\int_{-1}^{1} f(x) \cdot L_5(x)dx$, $\int_{-1}^{1} L_3(x) \cdot L_5(x)dx$ gde je $L_i(x)$ Ležandrov polinom stepena i. Prosleđena funkicija f može biti složena funkcija.

```
function integrali(f)

% Ako je funkcija prosleđena kao string, tada moramo kreirati inline
% objekat (ne možemo direktno anonimnu funkciju)

% Ako je funkcija prosleđena kao anonimna, onda je direktno možemo
proslediti
% ugrađenoj funkciji integral()
f = inline(f);
g = @(x) sin(x);
L = legendre_poly(5);
```

```
disp('Vrednost integrala funkcije f(x) je:');
integral(@(x) f(x),-1,1)
disp('Vrednost integrala funkcije f(x)*sin(x) je:');
integral(@(x) f(x).*g(x),-1,1)
disp('Vrednost integrala funkcije f(x)*L_5 je:');
integral(@(x) f(x).*polyval(L{6},x),-1,1)
disp('Vrednost integrala funkcije L_3*L_5 je:');
integral(@(x) polyval(L{4},x).*polyval(L{6},x),-1,1)
%quad(@(x) polyval(L{4},x).*polyval(L{6},x),-1,1)
% U prethodnim verzijama MATLAB-a quad() se koristio umesto integral()
```

15. zadatak - domaći

- Napisati M-fajl $legendre.m$ sa funkcijom $L = legendre(n)$ koja kao rezultat vraća Ležandrov polinom L stepena n na intervalu $[-1, 1]$.
- Napisati M-fajl $polinom.m$ sa funkcijom $P = polinom(n, m)$ koja kao rezultat vraća polinom P dobijen preko formule $P(x) = (1 - x^2)\frac{d^m}{dx^m}L_n(x)$, gde je $L_n(x)$ Ležandrov polinom stepena n na intervalu $[-1, 1]$.
- Napisati M-fajl $integral.m$ sa funkcijom $I = integral(n, m, tol)$ koja sa tačnošću tol približno određuje i kao rezultat vraća vrednost integrala $\int_{-1}^{1} P(x)e^x\,dx$. Integral računati korišćenjem uopštene Simpsonove formule. Polinom $P(x)$ je polinom dobijen pod (2).

  test primer

```
>> L=legendre(5)

L =

    7.8750         0   -8.7500         0    1.8750         0

>> P=polinom(5,3)

P =

 -472.5000         0  525.0000         0  -52.5000

>> I = integral(5,3,1e-4)

I =

   77.0221

Provera:
>> integral(@(x) polyval(P,x).*exp(x), -1,1)

ans =

   77.0222
```

Nelinearne jednačine

Nelinearne jednačine

19. zadatak

Neka je funkcija zadata eksplicitno komandnim M-fajlom $funkcija.m$

$funkcija.m$

```
f = @(x) cos(x) - x;

% Test: Njutn(0,0,1e-5)
% Tačna vrednost: fzero(f,0)
% Za poziv Njutn(0,0,1e-500) biće dostignut maksimalan broj iteracija
```

- Napisati M-fajl sa funkcijom $x = Njutn(x0, y, tol)$ koja za unete argumente $x0, y$ i tol vraća rešenje jednačine $f(x) = y$ (gde je $x0$ početna vrednost iterativnog procesa) izračunato Njutnovom metodom sa tačnošću tol . Kriterijum zaustavljanja je $|x_n - x_{n-1}| < tol$. Broj iteracija ograničiti na 100. U slučaju da je dostignut maksimalan broj iteracija štampati odgovrajuću poruku. (Pretpostavka je da su ispunjeni svi uslovi za primenu metode.)

```
function x = Njutn(x0, y, tol)

  % Ulazni argumenti su:
  %     x0 - početna vrednost iterativnog niza
  %     y - desna strana jednačine f(x)=y
  %     tol - tačnost
  % Izlazni argument:
  %     x - vrednost argumenta za koju je f(x)=y

% Pomoćni fajl:
%     funkcija.m - sadrzi anonimnu funkciju f

funkcija;

% Njutnova metoda rešava jednačinu oblika F(x)=0
f = @(x) f(x) - y;

F = sym(f); % konvertujemo funkciju u simboličku
F1 = diff(F); % izvod simboličke funkcije
```

```
f1 = matlabFunction(F1); % konvertujemo simboličku funkciju F1 u anonimnu funkciju

iter = 0;
while iter < 100
    x1 = x0 - f(x0) / f1(x0);
    iter = iter + 1;
    if abs(x1-x0) < tol
           break;
    end
    x0 = x1;
end

x = x1;
if iter == 100
    disp('Dostignut je maksimalan broj iteracija.')
end
```

- Napisati M-fajl $tablica.m$ u kome su zadati vektori Y i x_0 iste dužine n , kao i vrednost tol. Pretpostavka je da su elementi vektora Y različiti. U M-fajlu se formira tablica $(X_1, Y_1), \ldots, (X_n, Y_n)$ gde su $X_i, i = 1, \ldots, n$ rešenja jednačina $f(X_i) = Y_i$ dobijena korišćenjem funkcije iz prethodne tačke. Vektor x_0 sadrži odgovarajuće početne vrednosti za iterativni proces. Tablicu štampati u komandnom prozoru u formatu $X : X(1)\ X(2) \ldots X(n)\ Y : Y(1)\ Y(2) \ldots Y(n)$.

```
% vrednosti su izabrane proizvoljno
Y = 1:5; % desna strana jednačine f(x)=y
n = length(Y);
tol = 10^(-5); % tolerancija
x0 = zeros(1, n); % početne vrednosti
X = zeros(1, n); % koreni

for i = 1:n
    X(i) = Njutn(x0(i), Y(i), tol);
end

% ugrađena funkcija num2str(x,n) konvertuje broj x u string
% i zapisuje ga sa n značajnih cifara
disp(['X: ', num2str(X, 5); 'Y: ', num2str(Y, 5)])
```

- Napisati M-fajl $vredfunk.m$ sa funkcijom $y = vredfunk(x)$ koja vraća vrednost Lagranžovog interpolacionog polinoma u tački x dobijenog korišćenjem svih vrednosti iz formirane tablice. (Niz čvorova $x_i, i = 1, \ldots, n$ ne mora biti rastući)

```
function y = vredfunk(x)
% Ulazni argument: x - tačka u kojoj se računa vrednost funkcije
% Izlazni argument: y - približna vrednost funkcije
% Test: vredfunk(-3)

  tablica;
  y = 0;
  for i = 1:n
      p = 1;
      for j = 1:n
          if j ~= i
              p = p * (x - X(j)) / (X(i) - X(j));
```

```
            end
        end
        y = y + p * Y(i);
    end
```

20. zadatak

Neka je funkcija $f(x)$ zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, \ldots, x_n]$ i $F = [f_1, \ldots, f_n]$ (od kojih je prvi strogo rastući) za tu tablično zadatu funkciju. Tablica ne mora biti ekvidistantna.

$tablica.m$

```
% --------- I test --------------
X = linspace(0, 4, 5);
F = X - 1;
% -------- II test ---------------
% X = linspace(0,2,6);
% F = X.^3-exp(X);
% %F = exp(X).*sin(X)-cos(X);
```

- Napisati M-fajl $funk.m$ sa funkcijom $y = funk(x)$ koja najpre formira Njutnov interpolacioni polinom sa podeljenim razlikama na osnovu svih vrednosti vektora X i F iz fajla $tablica.m$, a zatim vraća vrednost formiranog polinoma za ulazni argument x.

```
function y = funk(x)

tablica;

n = length(X);
pr = zeros(n, n-1);

for i = 1 : n-1
    pr(i, 1) = (F(i+1) - F(i)) / (X(i+1) - X(i));
end

for j = 2 : n-1
    for i = 1 : n-j
        pr(i, j) = (pr(i+1, j-1) - pr(i, j-1)) / (X(i+j) - X(i));
    end
end

[X', F', pr]

% Konstruišemo interpolacioni polinom
polinom = [F(1)];
r = [1];

for i = 1 : n-1
    r = conv(r, [1, -X(i)]); % množimo polinome
    % svaki sledeći sabirak je polinom stepena za jedan većeg od prethodnog
    % dodavanjem nule na početak, omogućavamo sabiranje ova dva polinoma
    polinom = [0, polinom] + r * pr(1, i);
end
y = polyval(polinom, x);
```

- Napisati M-fajl $polov.m$ sa funkcijom $x = polov(tol)$ koja na intervalu $[x_1, x_n]$ računa i vraća rešenje jednačine $funk(x) = 0$ metodom polovljenja intervala sa tačnošću tol . Funkcija treba da proveri da li su uslovi za primenu metode polovljenja intervala ispunjeni, da prekine program i vrati poruku ukoliko nisu. Prvi i poslednji element vektora $X(x_1$ i $x_n)$ dobijaju se pozivanjem fajla $tablica.m$

```
function x = polov(tol)

tablica;
a = X(1);
b = X(end);

if (funk(a) * funk(b) > 0)
    error('Funkcija ne menja znak na posmatranom segmentu')
end

if funk(a) == 0
    x = a;
else
    if funk(b) == 0
        x = b;
    end
end

if (funk(a) ~= 0 && funk(b) ~= 0)
    iter = 0;
    while (abs(a-b) > tol)
        x = (a + b) / 2;
        iter = iter + 1;
        if funk(x) == 0
            break
        else
            if funk(a) * funk(x) < 0
                b = x;
          else
                a = x;
            end
        end
    end
    x=(a + b) / 2;
end
```

21. zadatak

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, \ldots, x_n]$ i $F = [f_1, \ldots, f_n]$ (od kojih je prvi strogo rastući) za tu tablično zadatu funkciju. Tablica ne mora biti ekvidistantna.

$tablica.m$

```
X = [0 0.3 0.55 0.60 0.98 1.07 1.27 1.33 1.6 2];
F = [0.2500    0.5455    0.7727    0.8146    1.0805    1.1272    1.2051
 1.2211    1.2496    1.1593];

%Drugi primer
%X=[1:10];
%F=log(X)+3;
%Test: nula(1,1e-4,20)
```

- Napisati M-fajl $funk.m$ sa funkcijom $y = funk(x)$ koja za unetu vrednost argumenta x vraća y, približnu vrednost funkcije u toj tački izračunatu pomoću Njutnovog interpolacionog polinoma sa podeljenim razlikama, koristeći sve vrednosti iz M-fajla $tablica.m$.

```
function y=funk(x)

tablica;
n=length(X);
pr=zeros(n,n-1);

for i=1:n-1
    pr(i,1)=(F(i+1)-F(i))/(X(i+1)-X(i));
end

for j=2:n-1
    for i=1:n-j
        pr(i,j)=(pr(i+1,j-1)-pr(i,j-1))/(X(i+j)-X(i));
    end
end

L=F(1);
p=1;
for i=1:n-1
    p=conv(p,[1,-X(i)]);
    L=[0 L]+p*pr(1,i);
end
y=polyval(L,x);

% Nije bilo neophodno konstruisati polinom (obzirom da nije naglašeno u
% zadatku)
```

- Napisati M-fajl $nula.m$ sa funkcijom $[x, briter] = nula(x_0, tol, iterM)$ koja računa i vraća x, rešenje jednačine $funk(x) = 0$ metodom proste iteracije sa tačnošću tol , kao i broj iteracija $briter$. Kriterijum zaustavljanja je: $|x_n - x_{n-1}| < tol$. Broj iteracija ograničiti na $iterM$ i ispisati poruku da li je zadovoljena tačnost ili je dostignut maksimalan broj iteracija. Vrednosti funkcije u tačkama koje se javljaju kroz iteracije računati pomoću funkcije $funk()$. Pretpostavka je da je funkcija na tom intervalu kontrakcija. Za početnu tačku iterativnog niza uzeti tačku x_0.

```
function [x, briter] = nula(x0, tol, iterM)

briter=0;

while (briter < iterM)
    x1 = funk(x0);
    briter = briter + 1;
```

```
    if abs(x0-x1) <= tol
        break
    end
    x0 = x1;
end

x = x1;

if briter == iterM
    disp('Dostignut je maksimalan broj iteracija!')
end

if abs(x0-x1) <= tol
    disp('Tacnost je zadovoljena!')
end
```

- Grafički prikazati, funkcijom $grafik(x_0, iterM)$u M-fajlu $grafik.m$, zavisnost brzine konvergencije od tačnosti tol ako se ona kreće od 10^{-4} do 10^{-3} sa korakom 10^{-4} . (Pod brzinom konvergencije podrazumeva se broj iterativnih koraka.)

```
function grafik(x0, iterM)

tol = 1e-4:1e-4:1e-3;
n = length(tol);
briter = zeros(1, n);

for i = 1 : n
    [x, briter(i)] = nula(x0, tol(i), iterM);
end

plot(tol,briter);
xlabel('Tolerancija');
ylabel('Broj iteracija');

title('Zavisnost brzine konvergencije od tacnosti tol');
```

22. zadatak

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 \ldots, x_n]$ i $F = [f_1, f_2 \ldots, f_n]$ (od kojih je prvi strogo rastući i ekvidistantan) za tu tablično zadatu funkciju.

$tablica.m$

```
X = 0:0.1:0.5;
F = exp(X) - 2 * (X - 1).^2;

% Test: [I,Y,x]=nula(0,0.5,1e-5,15) %zadovoljena tacnost tol
% Test: [I,Y,x]=nula(0,0.5,1e-5,4)  %zadovoljen max br. iteracija
% Test: [I,Y,x]=nula(0,0.5,1e-5,6)  %zadovoljena je tacnost i dostignut max
% br. iteracija
```

- Napisati M-fajl $Njutn1.m$ sa funkcijom $koef = Njutn1()$ koja vraća koeficijente I Njutnovog interpolacionog polinoma (po promenljivoj q), koristeći sve vrednosti iz tablice.

```
function koef = Njutn1()

tablica;
n = length(X);

KR = zeros(n, n-1); % tablica konacnih razlika
for i = 1:n - 1
    KR(i, 1) = F(i+1) - F(i);
end

for j = 2:n - 1
    for i = 1:n - j
        KR(i, j) = KR(i+1, j-1) - KR(i, j-1);
    end
end
%disp([X' Y' KR]);

% Konstruišemo I Njutnov interpolacioni polinom po promenljivoj q:
yk = F(1);
qk = [1, 0];
for j = 1:n - 1
    yk = [0, yk] + qk * KR(1, j) / factorial(j);
    qk = conv(qk, [1, -j]);
end
koef = yk;
```

- Napisati M-fajl $nula.m$ sa funkcijom $[I, Y, x] = nula(x_0, x_F, tol, iterM)$ koja računa i vraća nulu x tablično zadate funkcije iz $tablica.m$ metodom regula-falsi sa tačnošću tol . Kriterijum zaustavljanja je $|x_n - x_{n-1}| < tol$ gde su x_n i x_{n-1} dve uzastopne tačke dobijene metodom regula-falsi. Broj iteracija ograničiti na $iterM$ i ispisati poruku da li je zadovoljena tačnost ili je dostignut maksimalan broj iteracija. Za fiksiranu tačku u metodi uzeti x_F , a za početnu vrednost iterativnog procesa x_0 . Vrednosti funkcije u tački računati pomoću I Njutnovog interpolacionog polinoma dobijenog funkcijom $Njutn1()$. U vektore I i Y upisivati redni broj iteracije i vrednost funkcije u tačkama iz iterativnog niza, redom. (Pretpostavka je da su ispunjeni svi uslovi za primenu metode.)

```
function [I, Y, x] = nula(x0, xF, tol, iterM)

% Ulazni argumenti su:
%     x0 - početna vrednost iterativnog niza
%     xF - fiksirana tačka u metodi regula-falsi
%     tol - tačnost
%     iterM - maksimalni dozvoljeni broj iteracija
% Izlazni argumenti su:
%     I - vektor [1:iter] gde je iter broj iteracija
%         dok se ne zadovolji tačnost tol ili iterM
%     Y - vrednosti tablično zadate funkcije u tačkama iterativnog niza
%         izračunata pomoću I Njutnovog interpolacionog polinoma
%     x - nula tablično zadate funkcije

  tablica;
  h = X(2) - X(1);
```

```
koef = Njutn1(); % Koeficijenti I Njutnovog int. pol. po q

% Anonimna f-ja koja računa vrednost I Njutnovog polinoma u tački x
% Polinom je po q, zato nećemo direktno slati x, već odgovarajuću
% vrednost za q, tj. (x-X(1)/h))
f = @(x)polyval(koef, (x - X(1))/h);

Y = zeros(1, iterM); % Y ima najviše iterM elemenata
iter = 0;

while iter < iterM
    x1 = x0 - (f(x0) / (f(xF) - f(x0))) * (xF - x0); % regula-falsi
    iter = iter + 1;
    Y(iter) = f(x1);
    greska = abs(x1-x0);
    if greska < tol
        break
    end
    x0 = x1;
end

I = 1:iter;
Y = Y(1:iter);
if iter == iterM && greska < tol
    disp('Zadovoljena je tacnost i dostignut max br iteracija.');
elseif iter == iterM
    disp('Zadovoljen je maksimalan broj iteracija.');
else
    disp('Zadovoljena je tacnost tol');
end
x = x1;
```

23. zadatak

Neka je funkcija f zadata tablično M-fajlom $tablica.m$ koji generiše dva niza $X = [x_1, x_2 \ldots, x_n]$ i $F = [f_1, f_2 \ldots, f_n]$ (od kojih je prvi strogo rastući) za tu tablično zadatu funkciju. Tablica ne mora biti ekvidistantna.

$tablica.m$

```
X = 1:0.1:1.5;
F = sin(X) - X.^2 + 1;

% Test:  x=nula(1e-4,10)
```

- Napisati M-fajl $Lagranz.m$ sa funkcijom $L = Lagranz()$ koja vraća koeficijente Lagranžovog interpolacionog polinoma, koristeći sve vrednosti iz tablice.

```
function L = Lagranz()

tablica;

n = length(X);
L = zeros(1, n);
```

```
for i = 1:n
    p = 1;
    for j = 1:n
        if i ~= j
            p = conv(p, [1, -X(j)]) / (X(i) - X(j));
        end
    end
    L = L + p * F(i);
end
```

- Napisati M-fajl $nula.m$ sa funkcijom $x = nula(tol, iterM)$ koja računa i vraća nulu x tablično zadate funkcije iz $tablica.m$ metodom sečice sa tačnošću tol . Kriterijum zaustavljanja je $|x_n - x_{n-1}| < tol$, gde su x_n i x_{n-1} dve uzastopne tačke dobijene metodom sečice. Za prve dve iteracije uzeti krajeve intervala. Broj iteracija ograničiti na $iterM$ i ispisati poruku da li je zadovoljena tačnost ili je dostignut maksimalan broj iteracija. Vrednosti funkcije u tački računati pomoću Lagranžovog interpolacionog polinoma dobijenog pozivom funkcije $Lagranz()$. Funkcija treba da ispisuje redni broj iteracije i vrednost funkcije u tačkama iz iterativnog niza, redom. (Pretpostavka je da su ispunjeni svi uslovi za primenu metode).

```
function x = nula(tol, iterM)

%Ulazni argumenti su:
  %     tol - tacnost
  %     iterM - maksimalni dozvoljeni broj iteracija
%Izlazni argumenti su:
  %     x - nula tablicno zadate funkcije


  tablica;
  n = length(X);
  koef = Lagranz();
  f = @(x)polyval(koef, x);
  Y = zeros(1, iterM);
  iter = 0;

  x0 = X(1); %Xn-1
  x1 = X(n); %Xn

  while iter < iterM
      x2 = x1 - (f(x1) / (f(x0) - f(x1))) * (x0 - x1); % metoda sečice
      iter = iter + 1;
      Y(iter) = f(x2);
      greska = abs(x2-x1);
      if greska < tol
        break
      end
      x0 = x1;
      x1 = x2;
  end
  I = 1:iter;
  Y = Y(1:iter);

  if iter == iterM && greska < tol
```

```
      disp('Zadovoljena je tacnost i dostignut je maksimalan broj
iteracija.');
  elseif iter == iterM
      disp('Zadovoljen je maksimalan broj iteracija.');
  else
      disp('Zadovoljena je tacnost tol')
  end
  x = x2;

  disp(['I: ', num2str(I, 5)])
  disp(['Y: ', num2str(Y, 5)])
```

Sistemi linearnih jednačina

16. zadatak

- Napisati M-fajl $sistem.m$ sa funkcijom $x = sistem(A, b)$ koja metodom proste iteracije rešava sistem jednačina $Ax = b$. Broj iteracija fiksirati na 50.

  $sistem.m$

```
function x = sistem(A,b)

n = length(A); % Kada se length() primeni na matricu, vraća broj vrsti
A1 = zeros(n);
b1 = zeros(n,1);

% Potrebno je transformisati sistem Ax = b u oblik x = A1*x + b1
for i = 1:n
    A1(i,:) = -A(i,:) / A(i,i); % i-tu vrstu delimo sa -A(i,i)
    A1(i,i) = 0;                % na dijagonali su 0
    b1(i) = b(i) / A(i,i);      % i-ti el. vektora b delimo sa A(i,i)
end
disp('A1 =')
disp(A1)
disp('b1 =')
disp(b1)

% Početna aproksimacija rešenja je vektor b1
x = b1;

% Za dobijanje nove aproksimacije koristi se rekurentna formula: x_n =
A1*x_n-1 + b1
for i = 1:50
    x= A1 * x + b1;
end
```

- Napisati M-fajl $matrica.m$ sa funkcijom $[A, b, x] = matrica(broj, d)$ koja vraća kolonu b duzine d čiji su svi elementi jedinice, kvadratnu matricu $A_{d \times d}$ koja iznad dijagonale ima jedinice, po dijagonali ima $10 \cdot broj$, dok na prvoj poddijagonali ima vrednost $broj - 1$, na drugoj poddijagonali ima vrednost $broj - 2$, itd. kao i vektor x koji je rešenje sistema $Ax = b$ (koristiti funkciju $sistem()$ za nalaženje vektora x).

  $matrica.m$

```
function [A, b, x] = matrica(broj, d)
```

```
b = ones(d, 1); % vektor kolona dužine d sa svim jedinicama
ad = broj*10*ones(d,1);

% Formiramo matricu koja na glavnoj dijagonali ima vektor ad
A = diag(ad);

% Poziv diag(ad, 1) formiraće matricu koja na prvoj naddijagonali ima vektor
ad
% Poziv diag(ad, -1) formiraće matricu koja na prvoj poddijagonali ima
vektor ad

for i = 1:d-1
    ad = ones(d-i, 1);
    % popunjavamo matricu ispod (II sabirak) i iznad (I sabirak) glavne
dijagonale
    A = A + diag(ad*(broj-i),-1*i) + diag(ad,i);
end

disp('A =')
disp(A)
x = sistem(A, b);
disp('Resenje sistema je: ')
disp(x)
%Tačno rešenje sistema se može dobiti naredbom x = A\b
```

17. zadatak

- Formirati M-fajl $dominantna.m$ sa funkcijom $d = dominantna(A)$ koja proverava da li je zadata matrica A dijagonalno dominantna. Funkcija vraća vrednost 1 ako je matrica dijagonalno dominantna, inače vraća 0.

  $dominantna.m$

```
function d = dominantna(A)

n = length(A);
d = zeros(n, 1); % el. d(i) nosiće informaciju o ispunjenosti uslova u i-toj
vrsti

for i = 1:n
    % Proveravamo da li je suma apsolutnih vrednosti vandijagonalnih el. i-
te vrste
    % manja od apsolutne vrednosti el. na dijagonali
    if sum(abs(A(i, :))) - abs(A(i, i)) <  abs(A(i, i))
        d(i) = 1;
    else
        d(i) = 0;
    end
end
d = all(d==1);
% ili d = all(d);
```

- Formirati M-fajl $sistem.m$ sa funkcijom $[iter, x] = sistem(A, b, tol)$ koja nalazi rešenje sistema $Ax = b$ Gaus-Zajdelovom metodom pod uslovom da je matrica A dijagonalno dominantna. Inače ispisati poruku: "Matrica nije dijagonalno dominantna". Iterativni postupak se prekida kada za dve uzastopne iteracije važi $|x_k - x_{k-1}| \leq tol$. Program vraća rešenje x i broj iteracija $iter$.

  $sistem.m$

```
function [iter, x] = sistem2(A, b, tol)

% Prekidamo izvršavanje programa ukoliko matrica nije dijagonalno dominantna
if dominantna(A)==0
    error('Matrica nije dijagonalno dominantna');
end

n = length(A);
A1 = zeros(n);
b1 = zeros(n, 1);


for i = 1:n
    A1(i, :) = -A(i, :) / A(i, i);
    A1(i, i) = 0;
    b1(i) = b(i) / A(i, i);
end

q = norm(A1, inf); % uniformna norma matrice
tol = tol * (1-q) / q;

x0 = b1; % početna aproksimacija rešenja
x = x0;
iter = 1;

while 1
     % za nalaženje nove apoksimacije koordinate x_i tačnog rešenja
     % koristimo vrednosti koordianta x_j, j=1,...i-1 dobijenih iz iste
iteracije
     % zbog toga moramo koristiti for petlju i u njenom i-tom prolasku
     % računati vrednost koordinate x_i (razlika u odnosu na metodu proste
iteracije)
     for i = 1:n
         x(i) = A1(i, :) * x + b1(i);
     end
     iter = iter + 1;
     if all(abs(x - x0) <= tol) % kriterijum zaustavljanja
       break
     end
     x0 = x;
end

% Provera:
% disp('Tacno resenje:');
% disp(A\b);
```

18. zadatak

- Formirati M-fajl $LUdekompozicija.m$ sa funkcijom $x = LUdekompozicija(A, b)$ koja metodom LU dekompozicije vraća rešenje sistema $Ax = b$. Koristiti ugrađenu funkciju $lu()$.

  $LUdekompozicija.m$

```
function X = LUdekompozicija(A, b)

% Rešavamo sistem Ax = b metodom LU dekompozicije
% Na primer:
% A = [24.21 2.42 3.85; 2.31 31.49 1.52; 3.49 4.85 28.72];
% b = [1 0 0]';

n = length(b);
% Koristimo ugrađenu funkciju lu() koja vraća matrice L, U i P
[L U P] = lu(A);

disp('L =')
disp(L)
disp('U =')
disp(U);

% Sada se problem Ax = b svodi na rešavanje dva nova sistema
% L*y = b
% ----------------------------------------
% L(1,1)*y(1)                                     = b(1)
% L(2,1)*y(1) + L(2,2)*y(2)                       = b(2)
% L(2,1)*y(1) + L(2,2)*y(2) + L(2,3)*y(3)         = b(2)
% .....
% L(n,1)*y(1) + L(n,2)*y(2)+    ...    + L(n,n)*y(n)= b(n)
% ----------------------------------------
% U*x = y
% ----------------------------------------
% U(1,1)*x(1) + U(1,2)*x(2) +    ...    + U(1,n)*x(n) = y(1)
%               U(2,2)*x(2) +    ...    + U(2,n)*x(n) = y(2)
%                         .....
%                 U(n-1,n-1)*x(n-1) + U(n-1,n)*x(n) = y(n-1)
%                                      U(n,n)*x(n) = y(n)
% ----------------------------------------
%%%%%%%%%%%%%% rešavamo sistem L*y = b %%%%%%%%%%%%%
y = zeros(n, 1);
y(1) = b(1);

for i = 2:n
    y(i) = (b(i) - (L(i, :) * y));
end

disp('y =')
disp(y)
%%%%%%%%%%%%%% rešavamo sistem U*x = y %%%%%%%%%%%%%
X = zeros(n, 1);
X(n) = y(n) / U(n, n);

for i = n-1:-1:1
    X(i) = (y(i) - U(i, :) * X) / U(i, i);
end
```

  $LUdekomp.m$

```
function x=LUdekomp(A,b)

% kod za LU dekompoziciju (nije ugrađena funkcija, jedinice po dijagonali
kod U)

n = length(A);
L = zeros(n);
U = eye(n);

L(:, 1) = A(:, 1);
U(1, 2:n) = A(1, 2:n) / L(1, 1);

for i = 1:n
    for j = 1:n
        if j <= i
            L(i, j) = A(i, j) - (L(i, 1:(j-1)) * U(1:(j-1), j));
        else
            U(i, j) = (A(i, j)-L(i, 1:i-1) * U(1:i-1, j)) / L(i, i);
        end
    end
end
disp('L =');
disp(L);
disp('U =');
disp(U);

%%%%%%%%%%%%%% rešavamo sistem L*y = b %%%%%%%%%%%%%%
y = zeros(n, 1);
y(1) = b(1) / L(1, 1);
for i = 2:n
    y(i) = (b(i) - (L(i, 1:i-1) * y(1:i-1))) / L(i, i);
end
disp('y=')
disp(y)
%%%%%%%%%%%%%% rešavamo sistem U*x = y %%%%%%%%%%%%%%
x=zeros(n,1);
x(n)=y(n);
for j=n-1:-1:1
    x(j)=y(j)-(U(j,j+1:n)*x(j+1:n));
end
```

- Formirati M-fajl $inverzna.m$ sa funkcijom $inverzna = inverz(A)$ koja nalazi matricu A^{-1} korišćenjem funkcije iz fajla $LUdekompozicija.m$.

  $inverzna.m$

```
% primer: A = [7.05 0.11 0.13 0.15;
%              0.17 7.41 0.75 0.80;
%              0.97 1.02 8.11 0.11;
%              0.13 0.20 0.29 8.42];

function inverzna = inverz(A)

% dužina matrice A
[n, m] = size(A);

% E = jedinična matrica dimenzije nxn
```

```
E = eye(n);

inverzna = zeros(n);

for i = 1 : n
    % rešavamo n sistema: A*X(i) = E(i)
    % inverzna = [X(1) X(2) .. X(n)]
    % X(i) je i-ta kolona inverzne matrice
    % E(i) je i-ta kolona jedinične matrice
    inverzna(:, i) = LUdekompozicija(A, E(:, i));
end
```