1. Metoda bisekcije: Iščemo rešitev nelinearne enačbe $f(x) = 0$, z *metodo bisekcije*. Podani imamo krajišči začetnega intervala a, b in natančnost e. Funkcijski vrednosti v krajiščih za"etnega intervala$f(a)$ in $f(b) < 0$ sta različnega znaka. Program zaključimo, ko je $|f(x)| < e$ ali $|b - a| < e$.

   (a) Izračunamo vrednosti $f(a)$ in $f(b)$
   `IF` $f(a) \cdot f(b) < 0$ `THEN` nadaljujemo `ELSE` prekinemo program in javimo napako:

   ```
   error('bisekcija: f(a)*f(b)>=0')
   ```

   (b) `REPEAT` (zažetek zanke),

   (c) izračunamo vrednosti $x = \frac{a+b}{2}$ in $f(x)$,

   (d) `IF` $|f(x)| < e$ `THEN` zapustimo zanko in ponudimo rešitev x,

   (e) `IF` $f(a) \cdot f(x) < 0$ `THEN` $b = x$ `ELSE` $a = x$ in $f(a) = f(x)$,

   (f) `UNTIL` $|a - b| < e$, (ko je ipolnjen pogoj se zanka zaključi).

```
%[x,X]=bisekcija(f,a,b,e)
function [x,X] = bisekcija(f,a,b,e)
% vhod:
% f niz z definicijo funkcijskega predpisa,
% a in  b sta levo in desno krajisce zacetnega intervala,
% e, natancnost  abs(a-b) < e | abs(f((a+b)/2)) < e.
% izhod:
% x, resitev enacbe,
% X vektor zaporednih priblizkov.
```

2. Sekantna metoda: Iščemo rešitev nelinearne enačbe $f(x) = 0$, s *sekantno metodo*.
   Podana sta začetna približka a, b, natančnost e in maksimalno število iteracij n.
   Program zaključimo, ko je $|f(x)| < e$ ali $|b - a| < e$ oziroma je preseženo maksimalno število iteracij.

   (a) `IF` $f(a) < f(b)$, `THEN` zamenjaj a in b,

   (b) `FOR i=1 TO n` (začetek zanke) $x = b - f(b)\dfrac{a-b}{f(a)-f(b)}$,

   (c) postavimo $b = a$ in $a = x$,

   (d) `IF` $|f(x)| < e$ `OR` $|a - b| < e$ `THEN RETURN`,

   (e) `END` nadaljujemo z (b) dokler ni izčrpano maksimalno število iteracij.

```
%function [x,X] = sekant(f,a,b,e,n)
function [x,X] = sekant(f,a,b,e,n)
% vhod:
% f niz z definicijo funkcijskega predpisa,
% a, b zacetna priblizka.
% e, natancnost abs(a-b) < e | abs(f(x) < e
% maksimalno stevilo iteracij.
% izhod:
% x, resitev enacbe
% X vektor zaporednih priblizkov.
```

3. Metoda regula falsi: Iščemo rešitev nelinearne enačbe $f(x) = 0$ z *metodo regula falsi*. Podana sta začetna približka a in b tako, da sta $f(a)$ in $f(b)$ nasprotnega znaka in natančnost e. Program zaključimo, ko je $|f(x)| < e$ ali $|b - a| < e$.

   (a) `REPEAT` (začetek zanke)

   (b) $x = b - f(b)\dfrac{a - b}{f(a) - f(b)}$

   (c) `IF` $f(x)$ je nasprotnega znaka kot $f(a)$ `THEN` $b = x$ `ELSE` $a = x$.

   (d) `UNTIL` $|f(x)| < e$ `OR` $|a - b| < e$ (zanka traja, dokler ni izpolnjen pogoj)

```
%[x,X]=regula(f,a,b,e)
function [x,X] = regula(f,a,b,e)
% vhod:
% f niz z definicijo funkcijskega predpisa,
% a in  b sta levo in desno krajisce zacetnega intervala.
% e, natancnost abs(a-b) | abs(f(x)) < e
% izhod:
% x, resitev ena"cbe
% vektor zaporednih priblizkov.
```

4. Mulerjeva metoda: Iščemo rešitev nelinearne enačbe $f(x) = 0$, z *Mullerjevo metodo*. Trije začetni približki x_2, x_0, x_1 so podani v naraščajočem vrstnem redu.

   (a) izračunajmo funkcijske vrednosti $f_2 = f(x_2)$, $f_0 = f(x_0)$ in $f_1 = f(x_1)$,

   (b) `FOR` i=1 `TO` n

   (c) poiščimo koeficiente parabole $p(x)$ določene s tremi točkami (x_2, f_2), (x_0, f_0) in (x_1, f_1),

   (d) poiščemo oba korena kvadratne enačbe $p(x) = 0$,

   (e) izberimo koren, ki je najbližji točki x_0 in ga imenujmo x_r,

   (f) `IF` $x_r > x_0$ `THEN` $(x_2, x_0, x_1) := (x_0, x_r, x_1)$, `ELSE` $(x_2, x_0, x_1) := (x_2, x_r, x_0)$,

   (g) `IF` $|f(x_r)| < e$ `OR` $|x_2 - x_1| < e$ `THEN RETURN` x_r `END`.

   (h) `END` nadaljujemo z (b) dokler ni preseženo maksimalno število iteracij n.

```
%[x,X]=muller(f,a,b,c,e,n)
function [x,X] = muller(f,a,b,c,e,n)
% vhod:
% f niz z definicijo funkcijskega predpisa,
% a, b in c, a < b < c so zacetni priblizki,
% e natancnost, abs(a-c) < e | abs(f(b)) < e
% izhod:
% x=b, re"sitev ena"cbe
% vektor zaporednih priblizkov.
```

5. Newtonova metoda: Iščemo rešitev sistema nelinearnih enačb $\vec{f}(\vec{x}) = 0$, z *z Newtonovo metodo*. Podan je zčetni približek $\vec{x_0}$, natančnost e in maksimalno število iteracij n.
   Program zaključimo, ko je $|\vec{f}(\vec{x}_i)| < e$ ali $|\vec{x}_{i-1} - \vec{x}_i| < e$ oziroma je preseženo maksimalno število iteracij.

(a) `FOR i=1 TO n` (začetek zanke)

(b) `IF` $|\texttt{det}(\mathcal{J})| < e$ `THEN`

`error('newton: |det(J)|<e')`

`ELSE` $\vec{x}_i = \vec{x}_{i-1} - \mathcal{J}^{-1}(\vec{x}_{i_1}) \cdot \vec{f}(\vec{x}_{i-1})$,

(c) `IF` $|\vec{f}(\vec{x})| < e$ `OR` $|\vec{x}_i - \vec{x}_{i-1}| < e$ `THEN RETURN`,

(d) `END` nadaljujemo z (a) dokler ni izčrpano maksimalno število iteracij.

```
%[x,X] = newton(F,J,x,e,n)
function [x,X] = newton(F,J,x,e,n)
% vhod:
% F niz z definicijo funkcijskega predpisa,
% J niz z definicijo  Jakobijeve matrike,
% x zacetni priblizek.
% e, natancnost abs(dx) < e | abs(fx) < e
% maksimalno stevilo iteracij.
% izhod:
% x, resitev enacbe
% X vektor zaporednih priblizkov.
```

6. Newtonova metoda za polinome: Iščemo koren polinoma $p(x) = 0$, z *z Newtonovo metodo.*
Podan je vektor p s koeficienti polinoma, kot ga sprejema funkcija `polyval` v `MATLABU`, začetni približek a, natančnost e in maksimalno število iteracij n.
Program zaključimo, ko je $|p(x_i)| < e$ ali $|x_{i-1} - x_i| < e$ oziroma je preseženo maksimalno število iteracij. Odvod polinoma izračunamo s pomočjo `MATLABOVE` funkcije `polyder`.

(a) `FOR i=1 TO n` (začetek zanke)

(b) `IF` $|p'(x)| < e$ `THEN`

`error('newton: |p(x)|<e')`

`ELSE` $x_i = x_{i-1} - \dfrac{p(x_{i-1})}{p'(x_{i-1})}$,

(c) `IF` $|\vec{f}(\vec{x})| < e$ `OR` $|\vec{x}_i - \vec{x}_{i-1}| < e$ `THEN RETURN`,

(d) `END` nadaljujemo z (a) dokler ni izčrpano maksimalno število iteracij.

```
%
%[x,X] = newtonp(f,J,a,e,n)
function [x,X] = newtonp(p,a,e,n)
% vhod:
% p vektor koeficientov polinoma,
% a zacetni priblizek.
% e, natancnost abs(a-x) < e | abs(f(x)) < e
% maksimalno stevilo iteracij.
% izhod:
% x, resitev enacbe
% X matrika zaporednih priblizkov.
```

7. Napiši funkcijo ki bo prebrala zapise iz vhodne datoteke, kjer so zapisani ime metode, definicija funkcije in paramnetri. Funkcija pokliče ustrezno metodo in izpiše rezultate na izhodno datoteko. Če je oblika vhodne datoteke:

```
bisekcija x.^3-3*x.^2-x+3 0.2 2 1e-10
regula x.^3-3*x.^2-x+3 0.2 2 1e-10
sekant x.^3-3*x.^2-x+3 0.2 2 1e-10 10
newton x.^3-3*x.^2-x+3 3*x.^2-6*x-1 0.2 1e-10 10
newton [x(1,:).^3-3*x(1,:).*x(2,:).^2-1;3*x(1,:).^2.*x(2,:)-x(2,:).^3]
[3*x(1)^2-3*x(2)^2,-6*x(1)*x(2);6*x(1)*x(2),3*x(1)^2-3*x(2)^2] [1;1] 1e-10 10
```

je oblika izhodne datoteke:

```
Rezultati

Metoda bisekcije:

    f=x.^3-x.^2+3
    a=-1.5, b=-1
    e= 1e-10

  n         x         y

  1   -1.5000   -2.6250
  2   -1.0000    1.0000
  3   -1.2500   -0.5156
  4   -1.1250    0.3105
  5   -1.1875   -0.0847
  6   -1.1562    0.1173
  7   -1.1719    0.0174
  8   -1.1797   -0.0334
  9   -1.1758   -0.0079
 10   -1.1738    0.0047
 11   -1.1748   -0.0016
 12   -1.1743    0.0016
 13   -1.1746   -0.0000
 14   -1.1744    0.0008
 15   -1.1745    0.0004
 16   -1.1745    0.0002
 17   -1.1745    0.0001
 18   -1.1746    0.0000
 19   -1.1746    0.0000
 20   -1.1746    0.0000
 21   -1.1746   -0.0000
 22   -1.1746    0.0000
 23   -1.1746    0.0000
 24   -1.1746   -0.0000
 25   -1.1746   -0.0000
 26   -1.1746    0.0000
 27   -1.1746    0.0000
```

```
 28   -1.1746    0.0000
 29   -1.1746   -0.0000
 30   -1.1746    0.0000
 31   -1.1746    0.0000
 32   -1.1746    0.0000
 33   -1.1746   -0.0000
 34   -1.1746   -0.0000
 35   -1.1746    0.0000

Metoda regula falsi:

        f=x.^3-x.^2+3
        a=-1.5, b=-1
        e= 1e-10

  n         x         y

  1   -1.5000   -2.6250
  2   -1.0000    1.0000
  3   -1.1379    0.2316
  4   -1.1673    0.0469
  5   -1.1731    0.0092
  6   -1.1743    0.0018
  7   -1.1745    0.0004
  8   -1.1745    0.0001
  9   -1.1746    0.0000
 10   -1.1746    0.0000
 11   -1.1746    0.0000
 12   -1.1746    0.0000
 13   -1.1746    0.0000
 14   -1.1746    0.0000
 15   -1.1746    0.0000
 16   -1.1746    0.0000

Newtonova metoda :

        f=x.^3-x.^2+3
        x=-1.5
        e= 1e-10
        n=10

  n         x         y

  1   -1.5000   -2.6250
  2   -1.2308   -0.3792
  3   -1.1767   -0.0136
  4   -1.1746   -0.0000
  5   -1.1746   -0.0000
```

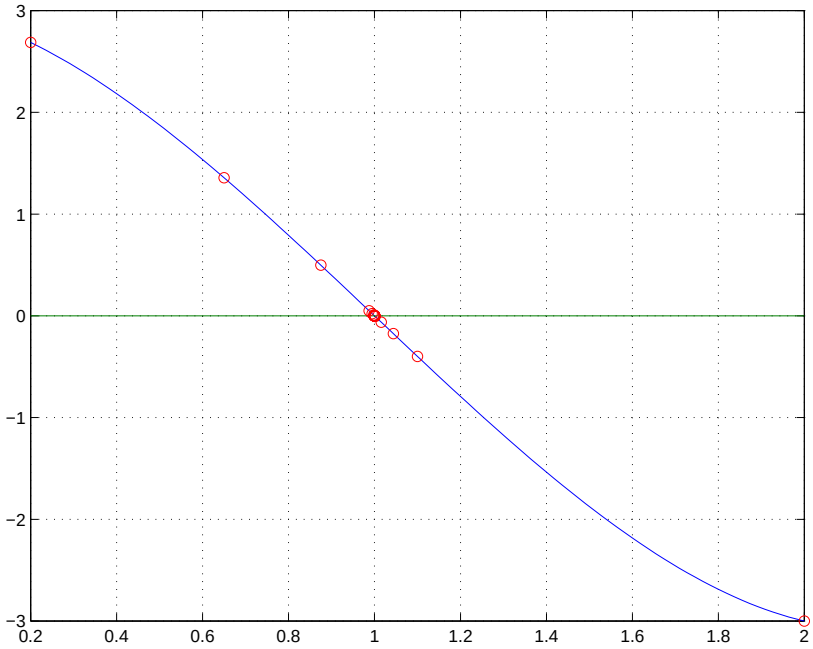
```
Newtonova metoda v dveh dimenzijah:

       f=[x.^3-3*x.*y.^2-1;3*x.^2.*y-y.^3]

   1        1.00000        1.00000       -3.00000        2.00000
   2        0.66667        0.50000       -1.20370        0.54167
   3        0.57884       -0.12747       -0.83427       -0.12606
   4        1.24696        0.31358        0.57108        1.43193
   5        1.00895        0.11368       -0.01203        0.34570
   6        0.98787        0.00384       -0.03600        0.01123
   7        1.00013       -0.00010        0.00040       -0.00029
   8        1.00000       -0.00000        0.00000       -0.00000
   9        1.00000       -0.00000       -0.00000       -0.00000
```

Naj se funkcija imenuje `nelinearne` sprejema naj imeni dveh datotek, ki so spravljena v spremenljivkah `metode` in `rezultati`. Pri vsaki metodi naj nariše sliko.

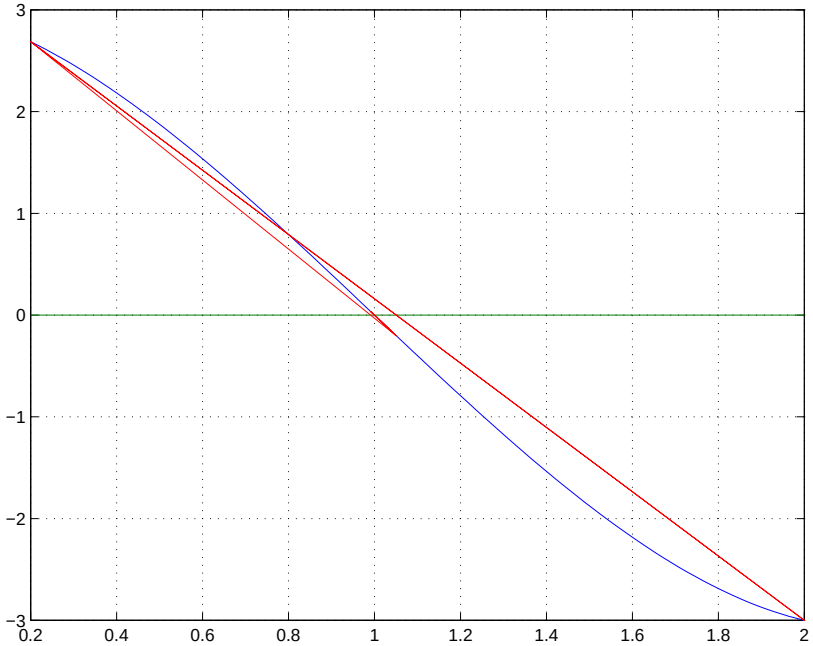
```
% Lupina za testiranje metod za resevanje nelinearnih enacb.
% nelinearne( metode, rezultati)
function nelinearne(metode,rezultati)
%
```

Slike ki spremljajo posamezno metodo

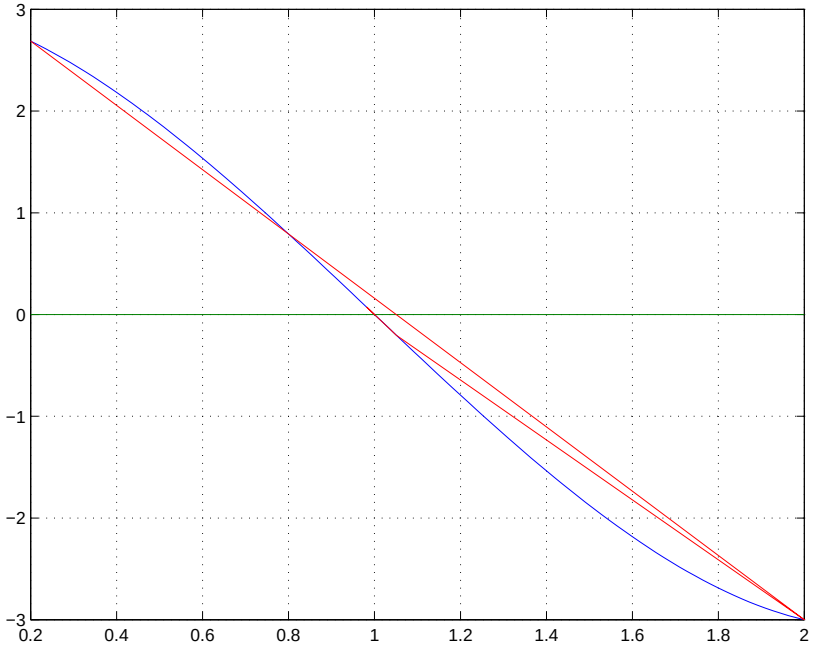
- slika k metodi bisekcije:



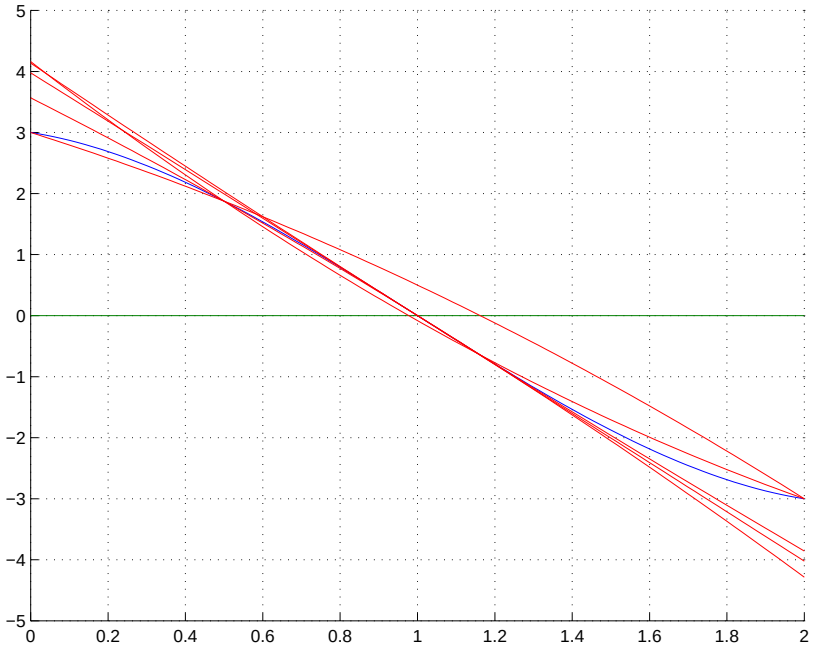
- slika k metodi regula:



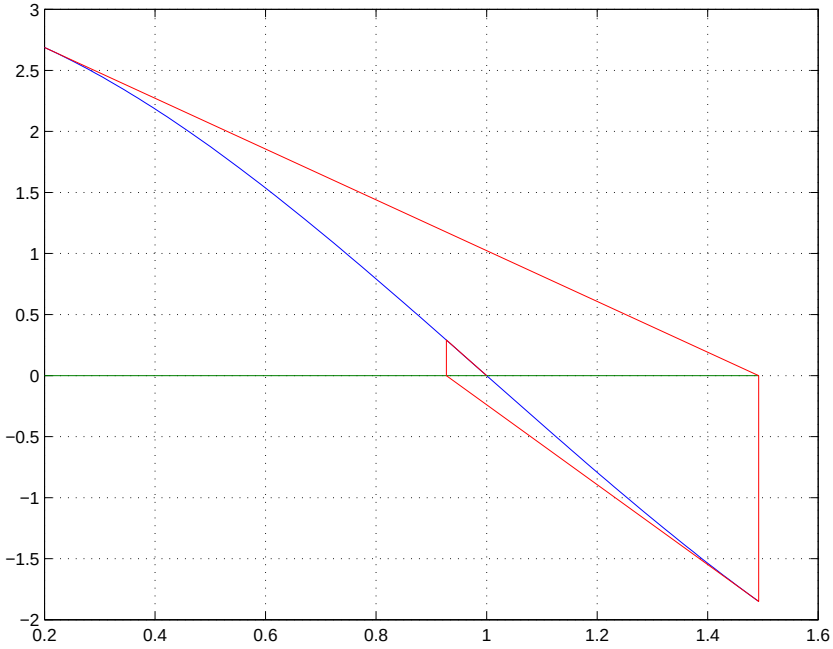
- slika k metodi sekant:



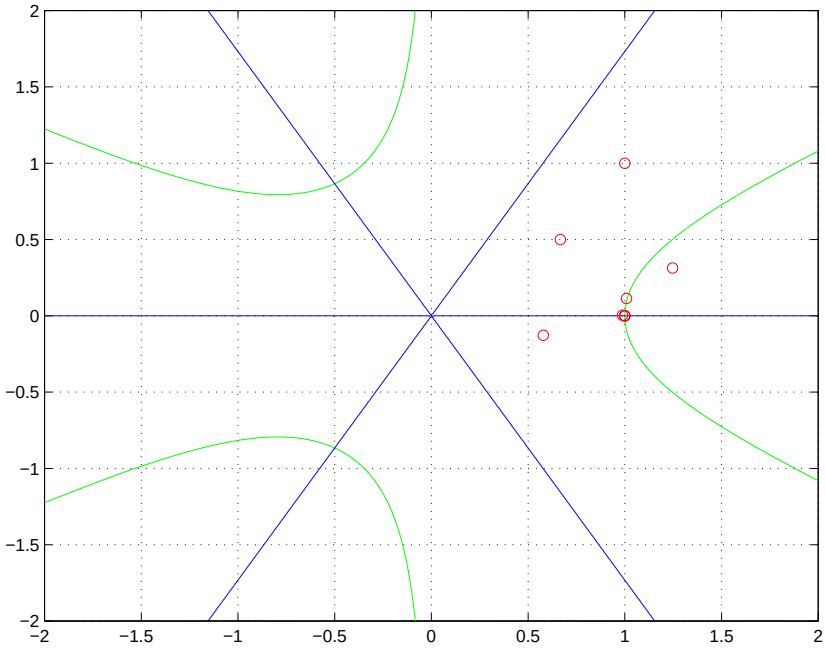
- slika k metodi muler:



- slika k newtonvi metodi:



- slika k newtonvi metodi v kompleksnem ali v dveh dimenzijah:



8. Poišči rešitve enačbe $z^m - 1 = 0$ v kompleksnem s pomočjo Newtonove iteracije. Izbiraj začetne približke na vozliščih mreže re(z) $\in$ x $= [-1 : 0.01 : 1]$ in im(z) $\in$ y $= [-1 : 0.01 : 1]$ in zapiši v matriko P na mesto $P(i, j)$ vrednosti $\{0, 1, \ldots, n\}$ glede na to ali je bilo izčrpano maksimalno število iteracij oziroma je zaporedje konvergiralo k prvemu, ... ali zadnjemu korenu. V matriko $N(i, j)$ zapisuj število iteracij. Torej $N(i, j)$ je enako številu iteracij newtonove metode pri začetnem približku $x(i) + i\, y(j)$. S pomočjo funkcij `image` oziroma `pclor` in `shading` nariši sliko. Maksimalno število iteracij n naj bo 100 in natančnost $e = 1e - 10$.

```
%[x,X] = julia(f,J,a,e,m,n)
function [P,N] = julia(x,y,e,m,n)
```

9. Jacobijeva iteracija.
Dan je sistem linearnih enačb$Ax = b$, zapisan v matrični obliki, kjer je A kvadratna matrika reda $n \times n$ in b stolpec desnih strani. Pravimo, da je matrika A strogo diagonalno dominantna po vrsticah, če velja:

$$|a_{ii}| > \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n$$

- Preveri, ali dani sistem lahko z menjavo **vrstic** in **stolpcev** preuredimo tako, da postane strogo diagonalno dominanten po vrsticah. Če je to mogoče, ga preuredi v ekvivalenten sistem, ki je strogo diagonalno dominanten po vrsticah.
- Reši ga z Jacobijevo iterativno metodo. Rešitev naj bo točna na 8 decimalnih mest.

Upoštevaj naslednji izrek o konvergenci Jacobijeve iterativne metode.
Dan naj bo sistem linearnih enačb $Ax = b$, kjer je A matrika reda $n \times n$ in b stolpec desnih strani. Zapišimo matriko A kot vsoto $A = D + (L + U)$, kjer je L spodnje trikotna matrika z ničlami na diagonali, D diagonalna matrika in U zgornje trikotna matrika z ničlami na diagonali.Če je A strogo diagonalno dominantna matrika po vrsticah, potem Jacobijeva iterativna metoda:

$$x_{r+1} = D^{-1}b - D^{-1}(L + U)r_n$$

konvergira za vsak začetni približek x_0. Za napako r-tega približka velja ocena

$$||x - x_r||_\infty \leq \frac{m}{1 - m}||x_r - x_{r-1}||_\infty,$$

kjer je $m = ||D^{-1}(L + U)||_\infty$ in x točna rešitev sistema.

```
%[x,j]=jacobi(A,b,x,e,n)
function [x,j] = jacobi(A,b,x,e,n)
% vhod:
% A je matrika sistema, b stolpec svobodnih clenov
% x, zacetni priblizek
% e natancnost,
% izhod:
% x, resitev enacbe
% j, stevilo iteracij.
```

10. Gauss-Seidlova iteracija.
Dan je sistem linearnih enačb$Ax = b$, kot v prejšnji nalogi. Preveri, ali dani sistem lahko preuredimo z menjavo **vrstic** in **stolpcev** tako, da postane strogo diagonalno dominanten po

vrsticah. Če je to mogoče, ga preuredi v ekvivalentnega in ga reši z Gauss-Seidlovo iterativno metodo. Rešitev naj bo točna na 8 decimalnih mest.
Upoštevaj naslednji izrek o konvergenci Gauss-Siedlove iterativne metode.
Dan naj bo sistem linearnih enačb $Ax = b$, kjer je A matrika reda $n \times n$ in b stolpec desnih strani. Zapišimo matriko A kot vsoto $A = (L + D) + U$. Če je A strogo diagonalno dominantna matrika po vrsticah, potem Gauss-Siedlova iterativna metoda:

$$x_{r+1} = (L + D)^{-1}b - (L + D)^{-1}Ur_n$$

konvergira za vsak začetni približek x_0. Za napako r-tega približka velja ocena

$$||x - x_r||_\infty \leq \frac{m}{1 - m}||x_r - x_{r-1}||_\infty$$

kjer je $m = ||(L + D)^{-1}U||_\infty$ in x točna rešitev sistema.

```
%[x,j]=siedl(A,b,x,e,n)
function [x,j] = siedl(A,b,x,e,n)
% vhod:
% A je matrika sistema, b stolpec svobodnih clenov
% x, zacetni priblizek
% e natancnost,
% izhod:
% x, resitev enacbe
% j, stevilo iteracij.
```

Primerjaj število iteracij, ki je potrebno za Jacobijevo in za Gauss-Siedlovo iteracijo pri isti natančnosti.

11. Metoda SOR.
Rešujemo sistem $Ax = b$, kjer je matrika A strogo diagonalno dominantna. Popravimo Gauss-Siedliovo iteracijsko shemo za reševanje tega sistema, na naslednji način:

$$x^* = (L + D)^{-1}b - (L + D)^{-1}Ur_n \quad x_{n+1} = \omega x^* + (1 - \omega)x_n, \quad \omega \in [1, 2)$$

Izberi šibko diagonalno dominantno tridiagonalno matriko. Vrednost elementov na glavni diagonali je 2, členi ob glavni diagonali pa so enaki -1. Poišči optimalno vrednost parametra ω, tako da bo število iteracij minimalmno. Če je $\omega = 1$ je to običajna Gauss-Siedlova metoda. Nariši graf števila iteracij v odvisnosti od relaksacijskega faktorja in reda sistema.

```
%[x,j]=SOR(A,b,x,e,n)
function [x,j] = SOR(A,b,x,w,e,n)
% vhod:
% A je matrika sistema, b stolpec svobodnih clenov
% x, zacetni priblizek, w relaksacijski faktor
% e natancnost,
% izhod:
% x, resitev enacbe
% j, stevilo iteracij.
```

12. Naj bo A kvadratna $2k + 1$ diagonalna matrika reda $n \times n$. To pomeni, da ima od nič različne elemente le na diagonali, na k zgornjih obdiagonalah ter na k spodnjih obdiagonalah.

Predpostavimo še, da je matrika A strogo diagonalno dominantna, torej

$$|a_{ii}| > \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n.$$

Primer: Matrika

$$A = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

je primer 3 diagonalne matrike reda 4×4, ki ustreza zgornjim pogojem.
Znano je, da za matrike z omenjenimi lastnostmi obstaja LU razcep in ga lahko izvedemo brez pivotiranja.

- Napiši program, ki za dano matriko z zgornjimi lastnostmi izračuna LU razcep.
- Preštej število potrebnih operacij (množenj in deljenj skupaj). Upoštevaj posebno zgradbo matrike in ne izvajaj nepotrebnih operacij. Matrika naj bo predstavljena samo z $2k + 1$ vektorji.
- Nariši graf odvisnosti števila operacij od števila neničelnih diagonal.

13. Reši linearni sistem enačb $Ax = b$. Če ima sistem neskončno rešitev, potem rešitev zapiši v obliki $[x, J]$, kjer je x katerekoli vektor, za katerega velja $A \cdot x = b$, medtem ko matrika J hrani v stolpcih bazo ničelnega podprostora, jedra, matrike A, $A \cdot J = 0$. Če sistem nima rešitev, potem poišči vse tiste x, ki minimizairajo izraz $(A \cdot x - b)(A \cdot x - b)$. Uporabljaš lahko vse `matlabove` ukaze. Poglej si ukaze `rref, null, orth` in $\backslash$.