Poročilo 1. domače naloge

Peter Rotar

Maj 2023

1. naloga: V programu Matlab je bilo potrebno sestaviti **kaznovalno metodo**, katera za problem Poissonove diskretne enačbe napravi utež točkam, katere se ne nahajajo v določenem prostoru Ω. Torej, problem se glasi:

$$\begin{cases} \Delta u(x,y) + k(x,y)u(x,y) = 1 & (x,y) \in [-1,1]^2 \\ u(x,y) = 0 & (x,y) \in \partial[-1,1]^2 \end{cases} \tag{1}$$

kjer velja

$$k(x,y) = \begin{cases} k & (x,y) \notin \Omega \\ 0 & (x,y) \in \Omega \end{cases} \tag{2}$$

in območje $\Omega = [-1,1]^2 - \{x^2 + y^2 < 1/10\}$. Za dani problem velikost $n \times n$, želimo sestaviti Poissonovo matriko velikosti $(n-2)^2 \times (n-2)^2$, ki ponazarja 5-točkovno shemo. Pred tem razširimo enačbo (1) s 5-točkovno shemo:

$$\begin{cases} 4u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i,j+1} - u_{i+1,j} + h^2 k(x_i,y_j)u_{i,j} = h^2 & (x_i,y_j) \in [-1,1]^2 \\ u_{i,j} = 0 & (x_i,y_j) \in \partial[-1,1]^2 \end{cases}$$

S tem znanjem lahko sestavimo Poissonovo matriko velikosti $(n-2)^2 \times (n-2)^2$ za točke $(x_i, y_i) \in (-1,1)$, kot je prikazano v ***kaznovalna_metoda.m***. Ponavadi bi morali še paziti na robne pogoje, vendar so v našem primeru nastavljeni na vrednost 0. Tako nam je uspelo dobiti razpršeni sistem $Ax = b$.

Sedaj lahko obravnavamo numerične rešitve v odvisnosti od n in k. Sprva se uporabi vgrajeno Matlab metodo ($u = A\backslash b$) in nato Jacobijevo metodo (ki se nahaja v datoteki ***Jacobi.m***), ki je izmed iterativnih najenostavnejša za implementirati, vendar v zameno potrebuje več korakov za konvergenco (hkrati lahko izvajamo vzporedno računanje). V datoteki ***Naloga1.m*** je izvedenih 6 preizkusov v odvisnosti od parametra n in k, kjer primerjamo vgrajeno Matlab metodo in Jacobijevo iteracijo.

n in k	norm(U_2-U_1)	max(max(abs(U_2-U_1)))
$n = 18,\ k = 50$	$3.798237831304833e^{-15}$	$5.551115123125783e^{-16}$
$n = 66,\ k = 100$	$9.483138912288737e^{-15}$	$5.134781488891349e^{-16}$
$n = 42,\ k = 21$	$5.866215168543380e^{-15}$	$5.273559366969494e^{-16}$
$n = 34,\ k = 2000$	$3.476488735357281e^{-15}$	$3.469446951953614e^{-16}$
$n = 38,\ k = 10^5$	$4.003145380910249e^{-15}$	$3.608224830031759e^{-16}$
$n = 44,\ k = 10^{12}$	$3.203229916317847e^{-14}$	$1.415534356397075e^{-15}$

Table 1: Preiskuzi vgrajene Matlab metode in Jacobijeve iteracije v odvisnosti od n in k. Matrika U_1 predstavlja rešitev z vgrajeno metodo. Matrika U_2 predstavlja rešitev z Jacobijevo iteracijo.

2. naloga: Napravimi moramo direktno Lanczsovo metodo, ki namestu LU razcepa uporabi QR razcep in se tako izogne kritičnim točkam. Sprva pričnemo enako kot D-Lanczos: poračunamo nove koeficienta α_k in β_k in zatem napravimo QR razcep, ki je podoben razcepu v *MINRES* metodi. Recimo, da je $k > 2$ ($k = 1, 2$ sta posebna primera, zato je njuna implementacija opisana na dnu naloge). Torej, imamo matrike T_{k-1}, R_{k-1} in Q_{k-1} ter koeficienta α_{k-1} in β_{k-1}. Seveda velja $T_{k-1} = Q_{k-1}R_{k-1}$, kjer je Q_{k-1} sestavljena iz Givensovih rotacij in R_{k-1} zgornje trikotna matrika. Napravimo nov korak iteracije, da dobimo α_k in β_k. Matriko T_k dobimo tako, da T_{k-1} razširimo z novo vrstico in stolpcem s primerno pozicijo α_k in β_{k-1}

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ & \beta_2 & \ddots & \ddots & & \\ & & \ddots & & & \\ & & & & \beta_{k-2} & \\ & & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ & & & & \beta_{k-1} & \alpha_k \end{bmatrix} = \begin{bmatrix} & & 0^{k-2} \\ & T_{k-1} & \beta_{k-1} \\ 0^{k-2} & \beta_{k-1} & \alpha_k \end{bmatrix}$$

Če pomnožimo razširjeno matriko Q_{k-1} z T_k, dobimo

$$\begin{bmatrix} Q_{k-1}^T & 0^{k-1} \\ 0^{k-1} & 1 \end{bmatrix} T_k = \begin{bmatrix} & & 0^{k-3} \\ & R_{k-1} & h_{k-1} \\ & & g_{k-1} \\ 0^{k-2} & \beta_{k-1} & \alpha_k \end{bmatrix}$$

Elementa h_{k-1} in g_{k-1} dobimo zato, ker zadnja Givensova rotacija matrike Q_{k-1}, t.j. $(k-2, k-1)$ zadene β_{k-1}, saj se ta nahaja v vrstici $k-1$. Dobimo ju na sledeči način:

$$\begin{bmatrix} h_{k-1} \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} c_{k-2} & s_{k-2} \\ -s_{k-2} & c_{k-2} \end{bmatrix} \begin{bmatrix} 0 \\ \beta_{k-1} \end{bmatrix} \tag{3}$$

Sedaj se želimo znebiti β_{k-1}, ki se nahaja v $k$ti vrstici. V tem primeru napravimo novo Givensovo rotacijo: vzemimo element nad β_{k-1}, torej zadnji element na diagonali R_{k-1}. Recimo mu $f_{k-1,k-1}$. Poračunamo rotacijo $(k-1,k)$:

$$c_{k-1} = \frac{f_{k-1,k-1}}{\sqrt{f_{k-1,k-1}^2 + \beta_{k-1}^2}} \tag{4}$$

$$s_{k-1} = \frac{\beta_{k-1}}{\sqrt{f_{k-1,k-1}^2 + \beta_{k-1}^2}} \tag{5}$$

in dobimo:

$$\begin{bmatrix} \tilde{f}_{k-1,k-1} & \tilde{g}_{k-1} \\ 0 & f_{k,k} \end{bmatrix} = \begin{bmatrix} c_{k-1} & s_{k-1} \\ -s_{k-1} & c_{k-1} \end{bmatrix} \begin{bmatrix} f_{k-1,k-1} & g_{k-1} \\ \beta_{k-1} & \alpha_k \end{bmatrix} \tag{6}$$

Končni rezultat nam da

$$R_{k-1,k}^T \begin{bmatrix} Q_{k-1}^T & 0^{k-1} \\ 0^{k-1} & 1 \end{bmatrix} T_k = R_{k-1,k}^T \begin{bmatrix} & & 0^{k-3} \\ & R_{k-1} & h_{k-1} \\ & & g_{k-1} \\ 0^{k-2} & \beta_{k-1} & \alpha_k \end{bmatrix} = \begin{bmatrix} & & 0^{k-3} \\ & \tilde{R}_{k-1} & h_{k-1} \\ & & \tilde{g}_{k-1} \\ 0^{k-2} & 0 & f_{k,k} \end{bmatrix} = R_k$$

kjer je razlika med $\tilde{R}_{k-1}$ in R_{k-1} zadnji element na diagonali. Sedaj lahko kot pri MINRES napravimo kratko rekurzijo in tako dobimo novi približek $x_k = x_{k-1} + \zeta_{k-1} * p_k$, kjer za ζ_k in p_k velja:

$$\zeta_k = \|r_0\|(-1)^{k-3} s_1 s_2 \cdots s_{k-2} s_{k-1} c_k \tag{7}$$

$$p_k = \frac{1}{f_{k,k}}(v_k - h_{k-1}p_{k-2} - \tilde{g}_{k-1}p_{k-1}) \tag{8}$$

To dobimo iz $P_k R_k = V_k$, $\|r_0\| Q_k e_1 = z_k$ in ostanek kot $\|r_k\| = \frac{\beta_k}{f_{k,k}} |s_1 \cdots s_{k-1}| \|r_0\|$.

Za $k = 1, 2$ v implementaciji sprva poračunamo α_1, β_1 in v_1, nato α_2, β_2 in v_2. Napravimo matriko T_2

$$T_2 = \begin{bmatrix} \alpha_1 & \beta_1 \\ \beta_1 & \alpha_2 \end{bmatrix}$$

in ročno napravimo QR razcep in dobimo

$$R_2 = \begin{bmatrix} f_{1,1} & g_1 \\ 0 & f_{2,2} \end{bmatrix}$$

poračunamo $p_1 = v_1 / f_{1,1}$ in $p_2 = (v_2 - g_1 p_1)/f2,2$. Končamo z novim približkom: $x_1 = x_0 + \|r_0\| p_1$ in $x_2 = x_1 + \|r_0\| c_1 p_2$. Upoštevajoč zgoraj opisanih točk (3) – (8) je sestavljena modificirana D-Lanczos metoda v datoteki ***QR_DLanczos.m***.

3. Naloga: Spomnimo se, če je matrika simetrična, mora biti tudi njena zgornja Hessenbergova matrika. To pa se zgodi v primeru, ko je H_k tridiagonalna in jo pišemo kar T_k. Seveda bi lahko za računanje uporabili metode, ki se ukvarjajo s splošnimi matrikami, vendar ti niso prirejeni temu, da optimalno izkoristijo lastnost tridiagonalne matrike T_k. Temu sledi večja prostorska in časovna zahtevnost, saj ti algoritmi hranijo več podatkov kot je potrebnih in računajo z ničelnimi vrednostmi. Zato sta v takem primeru optimalni metodi *MINRES* in *SYMMLQ*, kjer obe izkoristita tridiagonalst s kratko rekurzijo. Metodi sta prilagojeni takim problem, zato je časovna in prostorska zahtevnost manjša. V praksi lahko metode izboljšamo s predpogojevanjem. Naša matrika ne izpolnjuje pogojev za faktorizacijo Choleskega, torej (vsaj enostavno) ne moremo priti do predpogoja za *MINRES* in *SYMMLQ*. Metoda *GMRES* ne potrebuje, da je predpogoj s.p.d, zato lahko enostavno izvedemo nepopolni *LU* razcep. Hkrati *GMRES* nadgradimo s ponavljanjem, kar omogoča, da metode ne potrebuje vseh podatkov naenkrat. S tem pristopom sem dobil hitrejše izvajanje z nadgrajeno *GMRES* metodo. Slika spodaj prikazuje 10 ponovnih zagonov metod *MINRES*, *SYMMLQ* in *GMRES* na matriki $c-63$, za maksimalno število korakov 10^6, toleranca 10^{-6}. *GMRES* izvede cikel na vsakih 30 korakov. Preveril sem konvergence na 2000 iteracij. Hitro se razvidi, da modifi-

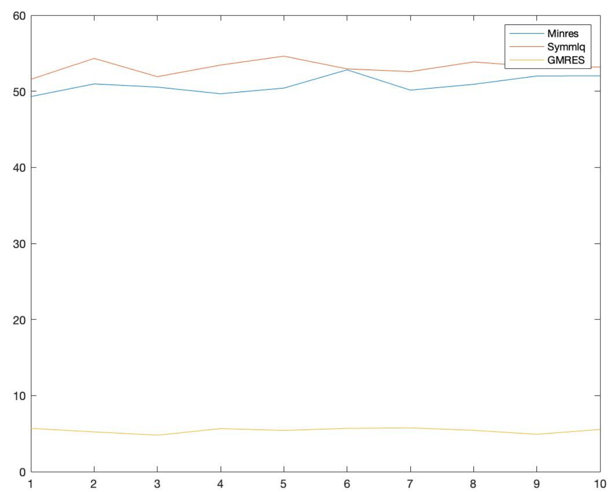

Figure 1: Čas delovanja metod na simetrični matriki v desetih iteracijah

ciran *GMRES* prevladuje, *MINRES* počasi pada. Metoda *SYMMLQ* bi za 10^5 korakov tudi skonvergirala.

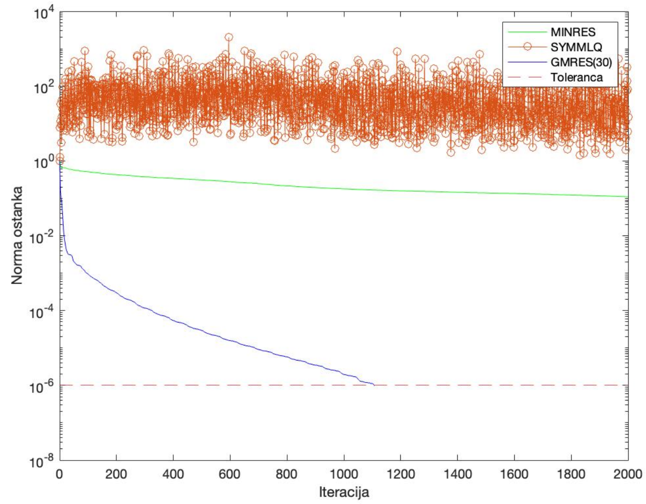

Figure 2: Metode za simetrično matriko

Za s.p.d matriko A, sta primerni tudi *MINRES* in *SYMMLQ*, vendar v primerjavi z njima metoda *konjugiranih gradientov* stopi korak naprej in izkoristi lastnost pozitivne definitnosti. Podobno kot prej, če je matrika A s.p.d, je matrika T_k tudi s.p.d. Kar pomeni, da LU razcep brez privotiranja zagotovo obstaja. S tem poznanstvom in pozitivne definitnosti lahko še lažje pridemo do novih vrednosti in tako prihranimo pri računski zahtevnosti v primerjavi z Lanczosovo metodo. Matrika *Gridgena* je s.p.d, zato za preverjanje konvergence uporabimo metode *MINRES*, *SYMMLQ*, modificiran *GMRES* in *metodo konjugiranih gradientov (pcg)*. *PCG* primerjamo z ostalimi sprva brez predpogojevanja (razen pri *GMRES*, nato s predpogojevanjem. Opazimo, da v obeh primerih metoda *PCG* prevladuje.

Sedaj preučujemo metode na matriki *RFdevice*. Ker je ta nesimetrična in ni pozitivno definitna, ne moremo uporabiti vseh zgornjih metod. Zato pa lahko vzamemo metode, kot so *GMRES*, *BiCG*, *QMR* in *BiCGSTAB*. V tem primeru bomo delali brez predpogojevanja, saj kakor bomo opazili so konvergence zgoraj predstavljenih metod hitre. Vzeli bomo toleranco 10^{-6} in število korakov 2000. Za vektor b sem vzel vektorji $(b(:,5), b(:,6), b(:,7), b(:,9))$. Pri vseh je konvergenca enaka. Kot je razvidno iz slike spodaj se konvergenca za *GMRES* ujema ravno s konvergenco metode *QMR*. Vse meritve so izvedene v datoteki ***Naloga3a.m***

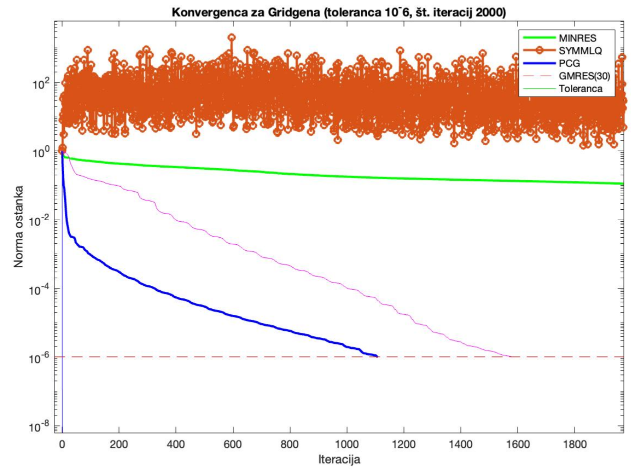

Figure 3: Metode za s.p.d matriko brez predpogojevanja

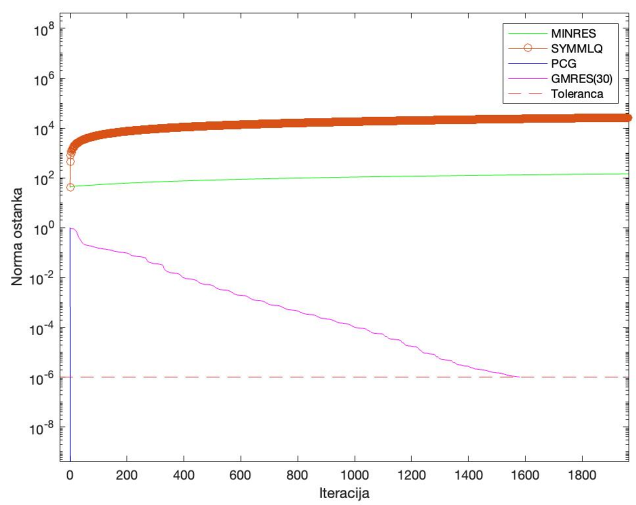

Figure 4: Metoda za s.p.d matriko s predpogojevanjem

S pomočjo empiričnih podatkov, sem lahko določil približno velikost optimalnega podprostora. Za vsako matriko sem uporabil podprostore velikosti 16, 32, 64, 100 in 128. Nato sem (kjer je bilo potrebno tudi uporabil predpogojevanje) poračunal hitrost konvergence za posamezno matriko. Pri *c-63* opazimo, da se *GMRES(128)* in *GMRES(100)* prekrivata. Kar nam da vedeti, da je optimalna zgornja meja med *GMRES(64)* in *GMRES(100)*. Z dodatnimi testi-

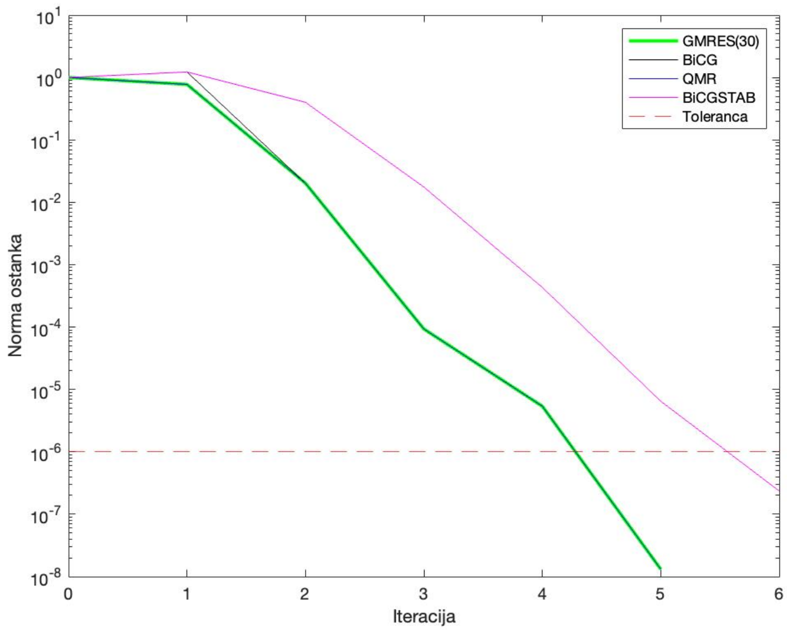

Figure 5: Metode za splošne matrike

ranji bi lahko hitro poiskali optimalen podprostor. Podobno lahko ocenimo za matriko *Gridgena*, da se velikost optimalnega podprostora nahaje v intervalu $(64, 100]$. Za matriko *RFdevice* opazimo, da je konvergenca vseh podprostorov približno enaka, torej vemo, da se velikost optimalnega podprostora nahaja v intervalu $[1, 16]$. Vse meritve so izvedene v datoteki ***Naloga3b.m***

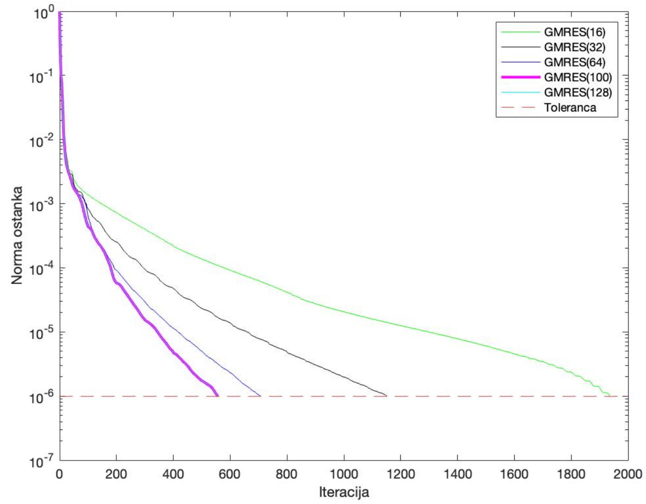

Figure 6: Konvergence podprostorov metode *GMRES* v matriki *c-63*

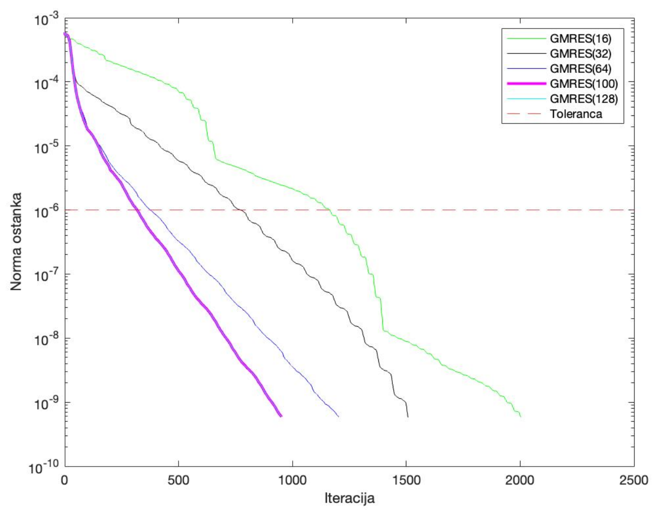

Figure 7: Konvergence podprostorov metode *GMRES* v matriki *Gridgena*

Za hitrejše računanje približka na tri decimalke natančno, sem uporabil iterativne motode, ki so se v prejšnih primerih pokazale najbolj optimalne. Za matriki *c-63* in *RFdevice* sem uporabil *GMRES(100)* in *GMRES(16)* (po zgledu iz optimalne velikosti podprostora) ter za matriko *Gridena* metodo *konjugi-*

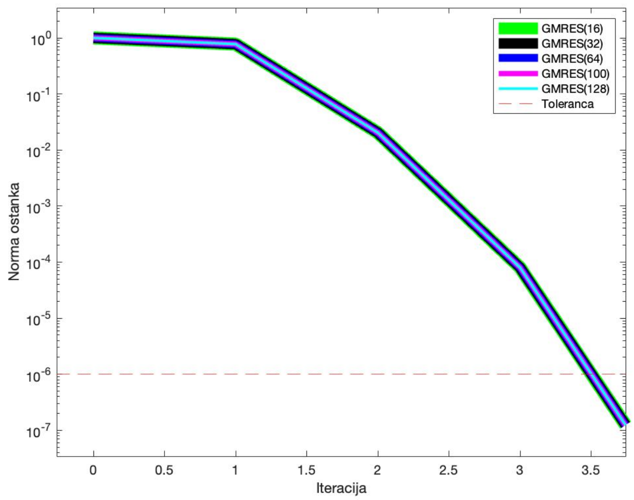

Figure 8: Konvergence podprostorov metode *GMRES* v matriki *RFdevice*

ranih gradientov. Za *c-63* se izkaže, da je vgrajena metoda primernejša tako po natančnosti, kakor po hitrosti. Prav tako se za *Gridena* izkaže, da je vgrajena metoda hitrejša. Nasprotno, za *RFdevice* se izkaže, da je vgrajena metoda $x = A\backslash b$ res natančnejša, vendar znatno počasnej. Odgovor na to se skriva v temu, da je matrika kompleksna in zato je računanje z vgrajeno metodo časovno potratno. Vse meritve so izvedene v datoteki ***Naloga3c.m***