Fakulteta za matematiko in fiziko

Iterativne numerične metode v lienarni algebri

1.domača naloga

Miha Avsec

21. december 2017

Naloga je bila samostojno reševana s programom Matlab 2016a.

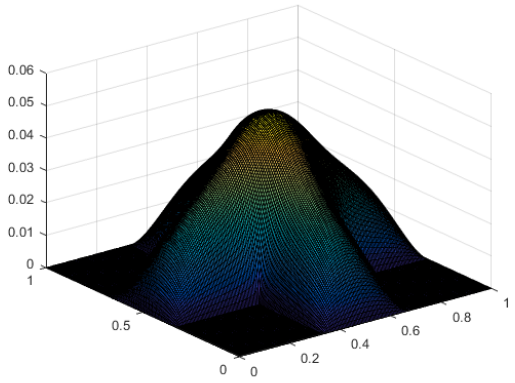
1.naloga

Prvo nalogo rešujemo s pomočjo vgrajenih funkcij numgrid in delsq. Najprej konstruiramo matriko delilnih točk, kjer so te točke v obliki plusa. To je narejeno s pomočjo funkcije numgridPlus. Funkciji podamo število točk na katere želimo razdeliti naše območje. Število točk mora biti v obliki $k*3+2$, saj imamo dve zunanji točki notranji del območja pa moramo enakomerno razdeliti na 3 dele. V funkciji numgridPlus zaporedoma kličemo funkcijo numgrid('S',n+2). kjer je $n = ((k * 3 + 2) - 2)/3$, tako da na primerna mesta postavimo kvadrate. Paziti moramo da prištejemo 2 točki saj moramo upoštevati, da numgrid('S',n+2) razdeli kvadrat na $n+2$ točk od katerih sta 2 robni. Končni integral nato izračunamo s funkcijo integral(N,risi), kjer je N število točk, ki jih podamo kot argument funkciji numgridPlus. Risi pa pove ali problem narišemo ali ne. Funkcija integral nastavi levo stran sistema s pomočjo funkcij numgridPlus in delsq. Za desno stran sistema pa vzamemo $2 \cdot h^2 \cdot e_1$. Tako dobimo U rešitev sistema v danem območju. Za izračunamo integrala množimo U z matriko $t \cdot t'$, kjer je t oblike

$$[1, 2, 2, \ldots, 2, 1].$$

Za boljši približek integrala na koncu interpoliramo vrednosti integrala glede na število delilnih točk in potem ta interpolant izračunamo v točki nič, ki naj bi predstavljala ničelno napako. Končni rezultat je tako 0.471081327583354, ki ga dobimo s klicem skripte nal1. Za vsako decimalko je potrebno število točk pomnožiti za faktor okoli 30. To pomeni, da bi za 6 točnih decimalk potrebovali okoli 729.000.000 delilnih točk.

Na spodnji sliki je prikazana slika območja, ki ga dobimo pri reševanju problema.



2.naloga

Naloga je rešena s funkcijo DLanczosPivot(A,b,x0,tol,maxit), ki dobi enake vhodne podatke, kot funkcija DLanczos. Funkcija deluje tako, da na vsakem koraku pogledamo ali je potrebno pivotiranje. V primeru, da pivotiranje ni potrebno deluje metoda enako, kot DLanczos le, da zaradi možnosti pivotiranja vektorja p, x računamo za korak nazaj. Če je pivotiranje potrebno v j-tem koraku ponastavimo vrednosti na sledeči način: Najprej zamenjamo vrstici. To naredimo tako, da $u(j-1) = starib, b(j-1) = a(j), d(j) = 0, d(j-1) = b(j)$, kjer je u vektor diagonalnih elementov matrike U iz LU razcepa, b predstavlja vektor prve nad diagonale U, d pa je vektor druge nad diagonale matrike U. Strarib predstavlja originalno vrednost $b(j-1)$, saj moramo paziti ali se je v prejšnjem koraku zgodilo pivotiranje. V tem primeru bi se spremenil tudi $b(j-1)$. Nato izračunamo $l(j)$, kjer je l vektor elementov matrike L, tako da delimo staro vrednost prejšnjega elementa na diagonali z novo vrednostjo elementa na diagonali. Popraviti moramo še elementa $u(j), b(j)$, kar naredimo na standarden način. Vrednost $z(j-1)$ nastavimo na 0, saj je le v tem primeru sistem $Lz = P||r_0 e_1||$ rešljiv. Obravnavati moramo še primer ali smo v koraku 2 ali ne. Če se nahajamo v koraku 2 smo pivotirali že prvo vrstico naše matrike, torej bo trenutni $z(2) = ||r_0||$, kar naredimo tudi, če smo v j-tem koraku a smo v vsakem dosedanjem koraku pivotirali. V nasprotnem primeru izračunamo nov element na sledeči način

$$z(j) = -sum(l(j - pivot : j). * z(j - 1 - pivot : j - 1)),$$

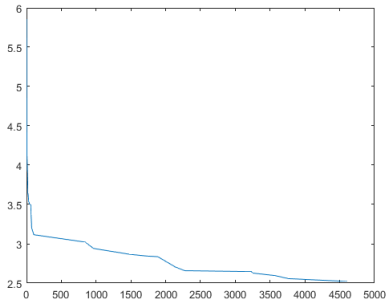
kjer moramo paziti da upoštevamo vse potrebne elemente vektorja l. V ta namen si shranjujemo ali pivotiramo ali ne, kot številko, kjer 0 pomeni, da ne pivotiramo, naravno število $n > 0$, pa pomeni da pivotiramo n-krat zapored. Ker vedno računamo p, x za prejšnji korak in se $z(j-1)$ ne bo več spreminjal je naš novi

$$p = 1/u(j-1) * (vp - b(j-2) * p - d(j-3) * pz),$$

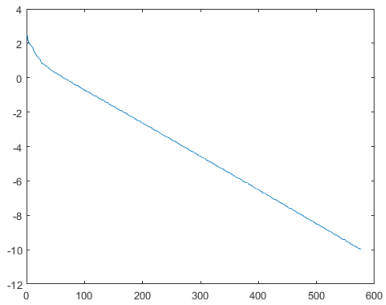
kjer je pz predzadnji vektor iz matrike P, x pa je enak

$$x = x + z(j-1)p.$$

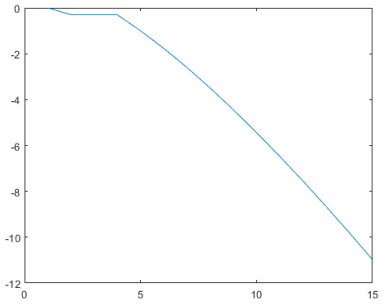
Prav tako moramo popraviti p, x še za tem ko končamo zanko. Metoda pri matrika DL3 divergira. Grafi konvergence pa so sledeči



Slika 1: Graf konvergence matrike DL1.



Slika 2: Graf konvergence matrike DL2.



Slika 3: Graf konvergence matrike DL4.

3.naloga

a) Matrika c-63

Matrika $c-63$ je simetrična, vendar ni pozitivno definitna.

Konvergenca gmres brez predpogojevanja je počasna, saj 500 korakov metode zadošča le za ostanek velikosti 10^{-2}. S predpogojevanjem pa lahko konvergenco pospešimo. Uporabimo lahko le nepopolni LU razcep saj matrika ni pozitivno definitna. A kljub predpogojevanju z delnim LU, kjer odstranimo elemente manjše od 0.1 metoda še vedno ne skonvergira do natančnosti 10^{-6} po 300 korakih. Optimalna izbira velikosti podprostora je 28, saj je tako časovna zahtevnost tam najmanjša.

Podobno, kot pri gmres je tudi pri minres konvergenca počasna. Po 500 korakih imamo ostanek velikosti 10^{-1}. Pri minres si ne moremo pomagati s predpogojevanjem ker delni LU razcep ni simetričen in pozitivno definiten. Metode pcg ne moremo uporabiti, ker matrika ni pozitivno definitna. Metoda qmr konvergira počasi. Po 500 korakih imamo ostanek velikosti 10^{-1}. Z enakim LU razcepom kot pri gmres je za željeni ostanek metoda potrebovala dobrih 700 korakov. A je za to potrebovala skoraj dvakrat toliko časa kot gmres(25). Metoda bicg brez predpogojevanja ni delovala. S predpogojevanjem pa je metoda prišla do približka, a le ta je bil slabši kot približek metode qmr izračunan v podobnem času. Metoda bicg-stab brez predpogojevanja ni prišla do rezultata. S predpogojevanjem, pa je metoda dala rezultat, a je bila konvergenca se vedno počasna.

Metoda symmlq nam v tem primeru ni dala odgovora, predpogojevanje pa nismo mogli izvesti, saj pri LU razcepu ne dobimo simetrične pozitivne matrike.

Najbolje se je na dani matriki odrezala metoda gmres s ponovnim zagonom.

Pričakovali bi, da se bo glede na obliko matrike najbolje odrezala metoda minres, a je konvergenca prepočasna, za predpogojevanje pa ne moremo uporabiti LU ali razcepa Choleskega. Glede na obliko pa se bi morala dobro obnesti tudi metoda gmres, ki je res najboljša za to matriko. Ce hočemo imete napako velikosti 10^{-3} ali več, pa metoda gmres(25) deluje hitreje kakor reševanje sistema v matlabu.

b) Matrika gridgena

Konvergenca te matrike je izjemno hitra že brez predpogojevanja. Uporabimo lahko vse metode, saj je matrika simetrična in pozitivno definitna. Pričakovali bi, da se bo najbolje odrezala metoda pcg zaradi strukture matrike, a temu ni tako. Najbolje od vseh metod se je odrezala metoda bicgstab, ki je za izračun potrebovala najmanj časa. S to metodo lahko tudi za veliko natančnost 10^{10} sistem rešimo hitreje, kot z vgrajeno matlabovo metodo $A\backslash b$.

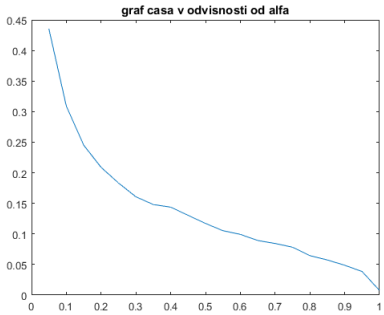
c) Matrika rajat25

Matrika *rajat*25 ni ne simetrična ne pozitivno definitna. Konvergenca je pri vseh metodah, ki jih lahko uporabimo počasna. Za predpogojevanje ne moremo uporabiti LU razcepa saj imamo na diagonali ničelne elemente, prav tako pa ne moremo uporabiti razcepa Choleskega. Na voljo imamo le metode gmres, bicg, qmr, bicgstab in symmlq. Metoda gmres je za večje podprostore delovala zelo počasi, napaka pa se ni veliko spremenila, zato je najboljša velikost okoli 10, odvisno od tega koliko decimalk potrebujemo. Symmlq je delovala najhitreje, a nam ni vrnila najboljšega ostanka. Najbolje se je odrezala metoda bicgstab če smo dopustili veliko število korakov, a se napaka ni dosti izboljšala. Za večjo napako okoli 10^{-3} pa je najboljša metoda qmr, kar bi tudi pričakovali glede na odločitveno drevo. Če smo zadovoljni s z ostankom okoli 10^{-3}, potem vse metode delujejo hitreje kakor reševanje sistema.

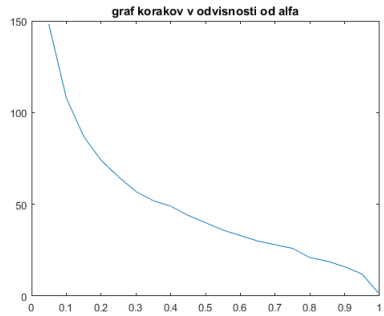
4.naloga

S pomočjo funkcije ocena določimo optimalen α, pri katerem želimo računati celoten inverz. Funkcija ocena za različne α poračuna pcg(A,b) in shrani

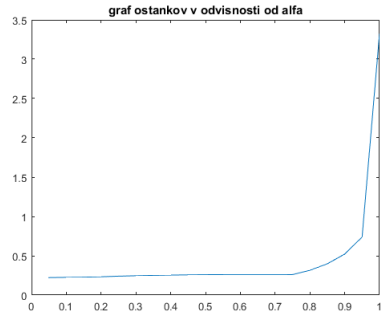
potrebno število korakov, ostanek metode in časovno zahtevnost metode. Na koncu za vse nariše grafe v odvisnosti od α.



Slika 4: Graf časa v odvisnosti od α.



Slika 5: Graf števila korakov v odvisnosti od α.



Slika 6: Graf ostankov v odvisnosti od α.

Vidimo, da se optimalen α nahaja okoli 0.7, saj so ostanki dobri časovna zahtevnost pa je tam majhna. Ko izberemo α poženemo funkcijo RegularizacijaInverz(B,test,20489,razlika,100), kjer je B funkcija množenja matrike pri danem α, test je matrika s katero hočemo množiti, 20489 je velikost matrike

train, razlika je zahtevana natančnost, 100 pa je število korakov posamezne metode pcg. Funkcija deluje tako, da za vsak enotski vektor izračuna inverzni stolpec matrike B, ki ga pomnoži z test' in ga shrani v začasni matriki. V vsakem koraku izračuna še razliko

$$||train * train' * inverzniStolpec(i) - e_i||_\infty.$$

Ostanek je potem največja od teh razlik. Končna rešitev se nahaja v datoteki *resitev.mat*, kjer je bil izbran $\alpha = 0.7$. Dodana pa je tudi rešitev *resitev*1.*mat*, kjer je bil izbran $\alpha = 0.2$.