Poročilo za 2. domačo nalogo
Numerična Linearna Algebra

Lenart Treven
Fakulteta za matematiko in fiziko
Oddelek za matematiko

9. junij 2018

1 Enostranska Jacobijeva metoda

V 1. nalogi smo implementirali enostransko Jacobijevo metodo. V datoteki *predznak.m* se nahaja funkcija, ki realnemu število priredi njegov predznak, s tem da številu 0 priredi predznak 1. V datoteki *zmnoziStolpce.m* se nahaja implementacija množenja z givensonovo rotacijo iz desne. V datoteki *onesidejacrot.m* se nahaja funkcija, ki v matriki G^TG uniči element na mestu (j, k), če le ni že dovolj majhen. Mejo, kdaj še uničimo element določa parameter *eps*. V datoteki *onesidejaccycle.m* je glavna funkcija, ki izvede ciklično enostransko Jacobijevo metodo. Metoda uničuje elemente, dokler v enem pregledu vseh elemenov, ne najde nobenega dovolj velikega, da bi ga uničila. Implementacija metod sloni na datotekah, ki jih ima prof. Bor Plestenjak objavljene na svoji spletni strani. Njegovi komentarji v kodi so večinoma ohranjeni. V našem primeru se metoda ustavi, ko uničimo 185 elementov. Te elemente uničimo v 7-ih ciklih. Približek za največjo singularno vrednost je 7.277868.

Tabela 1: Napake pri izračunu singularnih vrednosti

Singularna vrednost	Dejanska napaka	Maksimalna teoretična napaka
7.277868	$0.000e+00$	$1.346e-05$
1.963871	$4.441e-16$	$3.633e-06$
1.728863	$2.220e-16$	$3.198e-06$
1.415044	$4.441e-16$	$2.618e-06$
1.220059	$6.661e-16$	$2.257e-06$
1.160775	$2.220e-16$	$2.147e-06$
0.869198	$8.882e-16$	$1.608e-06$
0.831077	$6.661e-16$	$1.537e-06$
0.628127	$3.331e-16$	$1.162e-06$
0.489974	$2.220e-16$	$9.065e-07$

2 Deli in vladaj

Koda je organizirana v 5 različnih datotek. Opisi kaj se dogaja v funkcijah *psiFunkcija.m*, *sekul,m* in *sekular.m* so na začetku datotek. Datoteke so vzete iz predavateljeve spletne strani. Funkcija *deliinvladaj.m* temelji na datoteki iz predavateljeve spletne strani, vendar je popravljena tako, da naredi samo en korak metode *deli in vladaj*, nato pa poišče lastne vektorje in vrednosti s pomočjo vgrajene metode. Nadaljuje, kot je v navodilih. Funkcija vrne ortogonalno matriko q lastnih vektorjev matrike A , njej ortogonalno podobno matriko l, ter vektorja d, u in skalar rho, ki tvorijo sekularno enačbo.

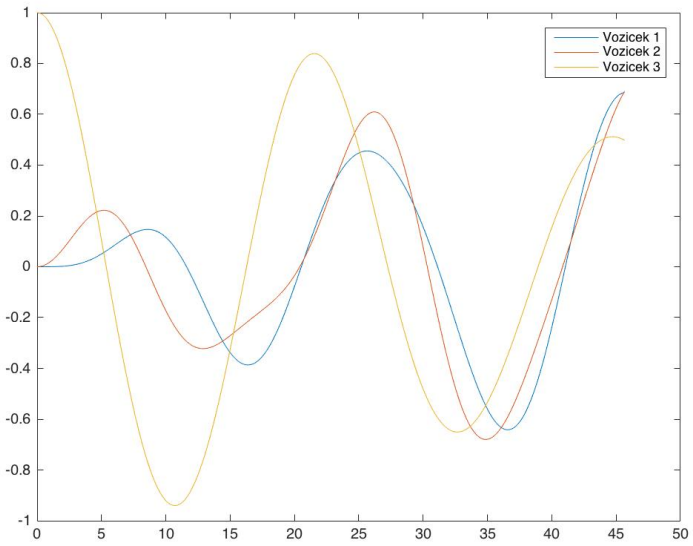
3 Vzmeti

Najprej zapišimo gibalne enačbe, ki veljajo za vozičke.

$$\begin{aligned} m_1\ddot{x}_1 &= -k_1x_1 + k_2(x_2 - x_1) \\ m_2\ddot{x}_2 &= -k_4x_2 - k_2(x_2 - x_1) + k_3(x_3 - x_2) \\ m_3\ddot{x}_3 &= -k_5x_3 - k_3(x_3 - x_2) \end{aligned}$$

To sestavimo v matrično enačbo in rešimo po zgledu iz profesorjevih prosojnic. Lastne frekvence so 0.5759, 0.3378 in 0.2752. Z upoštevnjem začetnih pogojev dobimo rešitev, ki je prikazana na sliki 1. Celotna rešitev je v datoteki *glavna.m*

Slika 1: Gibanje vozičkov na izbranem intervalu



4 Množenje v smereh

V datotekah *zmnozi.m* in *determinanta.m* se nahajata funkciji, kot sta podani v navodilih naloge. V datoteki *aliSmoEnaki.m* se nahaja funkcija, ki preveri, če ima vhodni vektor kakšno spremenljivko podvojeno. Funkcija *predznakPermutacije.m* vrne predznak permutacije, ki ga potrebujemo za izračun determinante. V datoteki *test.m* se nahaja algoritem, ki nadomešča n for zank, ki smo ga dobili na [1]. V spremenljivki *start_ranges* so shranjeni začetni indeksi for zank, v spremenljivki *end_ranges* pa so shranjene končne vrednosti for zank. Vse te datoteke uporabimo v *zmnoziEksperiment*, kjer implementiramo množenje tenzorja poljubnih dimenzij v neki smeri, ter v *determinantaEksperiment*, kjer s pomočjo tenzorjev lahko izračunamo determinanto za kvadratno matriko poljubne velikosti. Težava računanja determinante na tak način se pojavi že pri matrikah nizkega ranga, saj postane tezor X precej velik v prostorskem smislu.

Sedaj še razložimo, kako smo prišli do tenzorja X. Iščemo tak $X \in \mathbb{R}^{3\times3\times3}$, da bo za matriko $A \in R^{3\times3}$ veljalo:

$$\det(A) = X \times_1 a_1^T \times_2 a_2^T \times_3 a_3^T \tag{1}$$

Ker je $X \times_1 a_1^T \times_2 a_2^T \times_3 a_3^T \in \mathbb{R}^{1\times1\times1}$, ga lahko smatramo kot skalar. Posledično

velja formula:

$$X \times_1 a_1^T \times_2 a_2^T \times_3 a_1^T = (a_3^T \otimes a_2^T \otimes a_3^T)\,\mathrm{vec}(X)$$

Od tu je jasno, da če želimo, da velja formula (1), mora veljati:

$$X_{ijk} = \begin{cases} 1, & (ijk) \text{ je soda permutacija} \\ -1, & (ijk) \text{ je liha permutacija} \\ 0, & sicer \end{cases}$$

Na idejno enak način lahko X posplošimo na poljubne velikosti tako, da ga lahko uporabimo za izračun determinante matrik višjih dimenzij., kar smo uporabili v datoteki *determinantaEksperiment*.

5 HOSVD

Pri tej nalogi smo najprej implementirali *razpri.m*, ki nam tenzor razpre v i-ti izbrani smeri. Druga nova datoteka pri tej nalogi pa je *hosvd.m*, ki za vhodne parametre sprejeme trorazsežen tenzor T, ter željene dimenzije njegovega jedra. Funkcija nam vrne jedro, ter matrike U_i z ortonormiranimi stolpci. Da implementacija ne porabi preveč spomina sem namesto matlabove funkcije *svd* uporabil *svds*. Spodaj je prikazano jedro G, ki ga dobimo kot aproksimacijo v $\mathbb{R}^{3\times3\times3}$.

$$G(:,:,1) = \begin{pmatrix} -223.4892 & -0.00034657 & 0.0086884 \\ -0.10114 & 0.68346 & -0.1926 \\ 0.022835 & -0.48615 & -0.27346 \end{pmatrix}$$

$$G(:,:,2) = \begin{pmatrix} -0.0040499 & 0.47046 & 1.2566 \\ -0.095096 & 0.60214 & 0.0045397 \\ 0.76209 & 0.65776 & -0.97128 \end{pmatrix}$$

$$G(:,:,3) = \begin{pmatrix} 0.01458 & -1.1196 & -0.76031 \\ -0.35662 & -0.45619 & 1.0198 \\ -0.20352 & 0.24436 & -0.34517 \end{pmatrix}$$

Literatura

[1] https://stackoverflow.com/questions/7186518/function-with-varying-number-of-for-loops-python

[2] https://www.fmf.uni-lj.si/∼plestenjak/Vaje/Primeri.htm