O LSQR solverju, predpogojevalcih in dinamični natančnosti

Jure Ravnik

13. avgust 2009

Povzetek

V poročilu poročam o razvoju blok-diagonalnega predpogojevalca za predoločene sisteme enačb in o novem algoritmu za dinamično določanje zahtevane natančnosti interativnega solverja med nelinearno zanko.

1 Reševanje predoločenih sistemov enačb

Naj bo A pravokotna matrika, x neznani vektor in b znan desna stran

$$Ax = b \tag{1}$$

Razpolagamo z dvema solverjema - direktnim MA49 in iterativnim Paige and Sounders [1]. Pseudo inverz predoločenega sistema je

$$A^{-1} = (A^T A)^{-1} A^T. \tag{2}$$

Idealen predpogojevalec pa

$$P^{-1} = \sqrt{(A^T A)^{-1}}. \tag{3}$$

Na sistemu (1) ga uporabimo takole

$$AP^{-1}Px = b \qquad \rightarrow \qquad AP^{-1}x' = b, \qquad x = P^{-1}x' \tag{4}$$

Če P^{-1} izračunamo v celoti, potem iterativni [1] solver potrebuje samo eno iteracijo do rešitve, rezultat pa je identičen rešitvi direktnega solverja. Izračun celotnega predpogojevalca je prezahteven, zato izračunamo samo blok-diagonalni del predpogojevalca, ki ima sledečo neničelno strukturo

$$P^{-1} = \begin{pmatrix} x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & x & x & 0 \\ 0 & 0 & 0 & x & x & x & 0 \\ 0 & 0 & 0 & x & x & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x \end{pmatrix} \tag{5}$$

Zaradi blok-diagonalne neničenlne strukture, je mogoče tako inverz kot tudi koren izračunati po blokih. Inverz blokov računamo s pomočjo rutine *gaussj* iz

[2]. Koren matrike $\sqrt{B}$, za katerega velja $B = \sqrt{B} \cdot \sqrt{B}$, pa računamo s pomočjo dekompozicije na lastne vrednosti (rutina *jacobi* iz [2]):

$$B = QDQ^T \qquad \rightarrow \qquad \sqrt{B} = Q\sqrt{D}Q^T, \tag{6}$$

kjer je D diagonalna matrika lastnih vrednosti. Če velja $B = A^T A$ potem so vse lastne vrednosti B realne in pozitivne in je koren mogoče izračunati.

Učinkovitost predpogojevalca je odvisna od širine bloka. Če je blok širok 1 imamo t.i. diagonalni predpogojevalec, ki ga je mogoče zelo hitro izračunati. Na drugi strani, če je blok enako velik kot cela matrika, rabi solver samo eno iteracijo.

1.1 Ugotovitve

- Ne glede na izbrani predpogojevalec, vrne iterativni solver, pri zahtevani natančnosti 10^{-15} popolnoma enako rešitev kot direktni solver. Če zahtevamo manjšo natančnost se rešitev iterativnega solverja ujema z rešitvijo direktnega solverja samo do zahtevane natančnosti.

- Če izračunano rešitev x pomnožimo z A in rezultat primerjamo z originalno desno stranjo ugotovimo, da se **ZELO SLABO** ujemata. Uniformna norma je reda velikosti 1!!! To mi je dalo misliti, da torej nima nobenega smisla preveč natančno reševati predoločene sisteme med nelinearnim iterativnim postopkom.

2 Dinamično spremenljiva natančnost LSQR solverja

Uporabil sem naslednji algoritem za spreminjanje zahtevane natančnosti iterativnega solverja ϵ med nelinearno zanko. Naj bosta ϵ_{max} in ϵ_{min} zgornja in spodnja meja, R razmerje in ϵ_{err} sprememba izbranega polja (v_x, v_y, v_z, ω_x , ω_y, ω_z, T) potem

- berem input datoteko ter določim največjo in najmanjšo natančnost solverja ter razmerje R. Priporočene vrednosti so:
  - $\epsilon_{max} = 10^{-3}$,
  - $\epsilon_{min} = 10^{-7}$,
  - $R = 10$.
- začetek novega časovnega koraka
- Prvi dve iteraciji naj velja $\epsilon = \epsilon_{min}$, nato po koncu druge iteracije postavimo $\epsilon = \epsilon_{max}$. To je potrebno zato, ker imam zanesljiv izračun konvergence šele po drugi iteraciji. Da sem na varni strani, torej prvi dve izračunam z največjo natančnostjo.
- Na koncu iteracije (od druge naprej), po izračunu konvergence, določimo nov ϵ takole
  - če $\epsilon_{err}/R < \epsilon$ potem $\epsilon = \epsilon_{err}/R$

– če je nov ϵ večji od starega, potem spremembe ne dovolim. To je zaradi možnosti, da konvergenca oscilira. Na sliki 6 se delovanje tega pogoja jasno vidi. Ko koda nekaj iteracij divergira, ostane natančnost solverja konstantna.

– če je nov ϵ manjši od ϵ_{min}, potem $\epsilon = \epsilon_{min}$

Fortran implementacija:

```
DO ∀ equations
  ϵ_old = ϵ
  IF (ϵ_err/R .LT. ϵ) THEN
    ϵ=ϵ_err/R
  ELSE
    ϵ=ϵ_max
  END IF
  IF (ϵ_old .LT. ϵ) ϵ = ϵ_old
  IF (ϵ .LT. ϵ_min) ϵ = ϵ_min
END DO
```

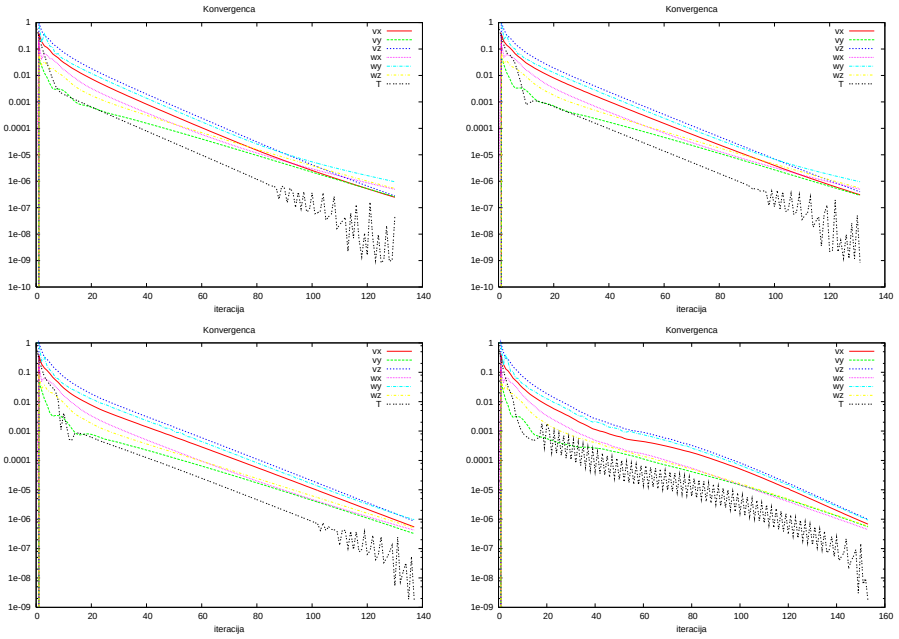
Algoritem je bil testiran - rezultati so v preglednici 1.

ϵ_{min}	ϵ_{max}	R	LSQR nit	čas LSQR [min]	Non-lin nit
Hotstrip, 46, $Ra = 10^3$, $25 \times 20 \times 7$					
10^{-7}	10^{-7}	-	103522	75	131
10^{-7}	10^{-3}	100	73632	55	132
10^{-7}	10^{-3}	10	57169	45.7	138
10^{-7}	10^{-3}	1	37875	35.5	154
Hotstrip, 46, $Ra = 10^4$, $25 \times 20 \times 7$					
10^{-7}	10^{-7}	-	148841	108	205
10^{-7}	10^{-3}	10	79906	66.3	207
Hotstrip, 46, $Ra = 10^5$, $25 \times 20 \times 7$					
10^{-7}	10^{-7}	-	759952	570	1235
10^{-7}	10^{-3}	10	410572	363	1255
10^{-7}	10^{-3}	1	202899	228	1179
10^{-7}	10^{-3}	0.1	150649	241*	1774

Tabela 1: Primerjava konvergence in števila potrebnih iteracij LSQR solverja za različne vrednosti razmerja R. * čas pri $R = 0.1$ je večji, ker je število nelinearnih iteracij večje in se je tako povečal čas za pripravo desnih strani.

V tabeli 1 vidimo, na primer pri $Ra = 10^3$, da število iteracij iterativnega solverja, ki je ob stalni natančnosti $\epsilon = 10^{-7}$ enako 103 tisoč, upade na več kot polovico. In ker večino časa porabimo prav v iterativnem solverju, je tudi prihranek časa podoben. Čas nelinearne zanke smo zmanjšali iz 75 na 35.5 minut! Pri $Ra = 10^5$ pa smo število iteracij zmanjšali iz 759 tisoč na 202 tisoč in čas iz 570 minut na 228 minut. Na slikah od 1 do 7 prikazujemo potek konvergence, zahtevane natančnosti solverja in števila iteracij solverja v odvisnosti od nelinearne iteracije.

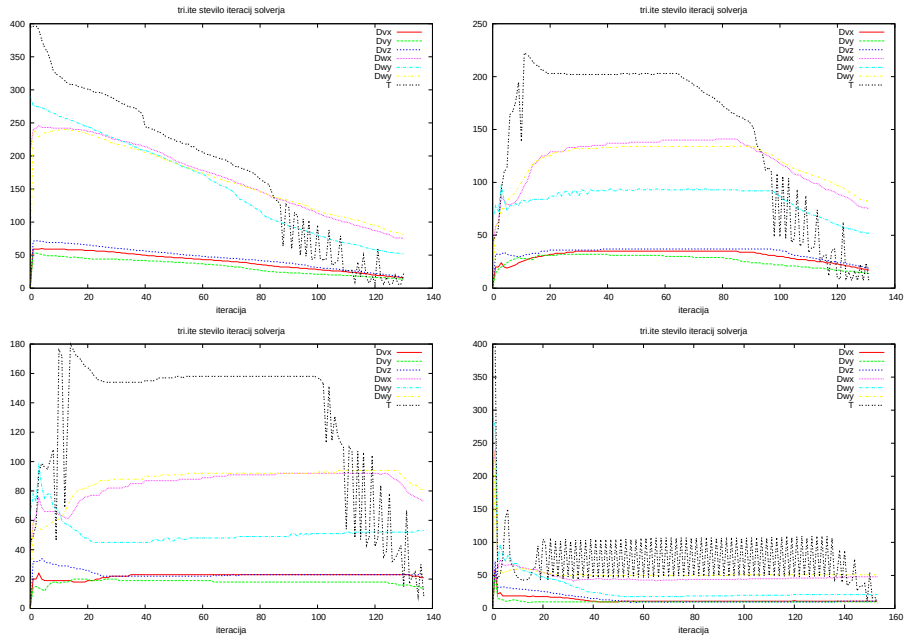
Na sliki 7 so prikazani konvergenca, število iteracij LSQR solverja in zahtevana natančnost solverja za primer $Ra = 10^5$ in $R = 0.1$. Tako majhen R



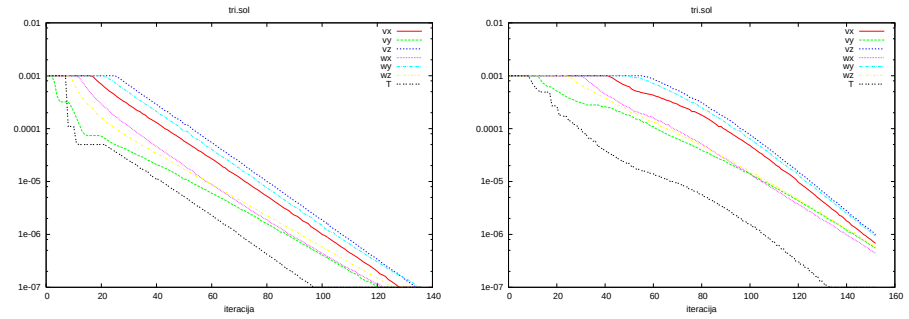
Slika 1: Primer $Ra = 10^3$. Konvergenca. Levo zgoraj konstantna natančnost $\epsilon = 10^{-7}$, desno zgoraj $R = 100$, levo spodaj $R = 10$, desno spodaj $R = 1$. Vidimo, da pri $R = 1$ temperatura začne oscilirati, navkljub temu je rezultat pravilen in pridobitev časa še vedno velika.

povzroči velike oscilacije v konvergenci zaradi zelo majhne zahtevane natančnosti solverja.

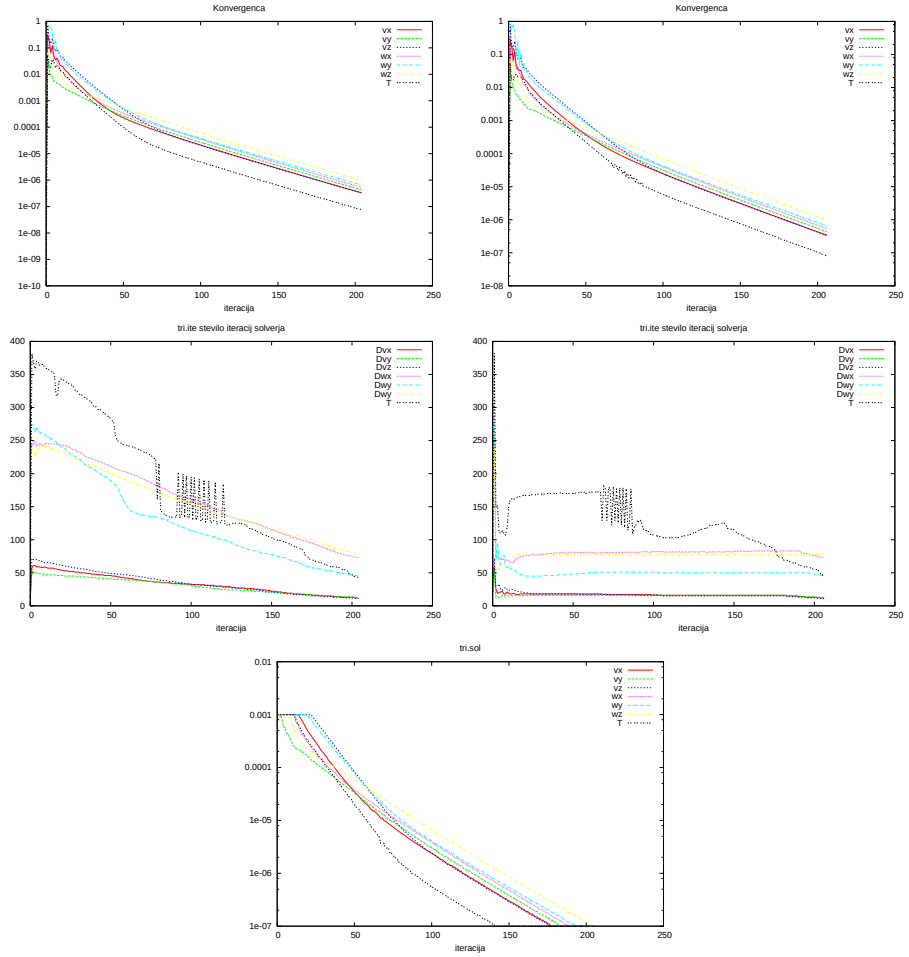
Na podlagi rezultatov testiranja lahko zaključimo, da naj se vrednost R giblje med $1 \leq R \leq 100$. Priporočamo uporabo $R = 10$.



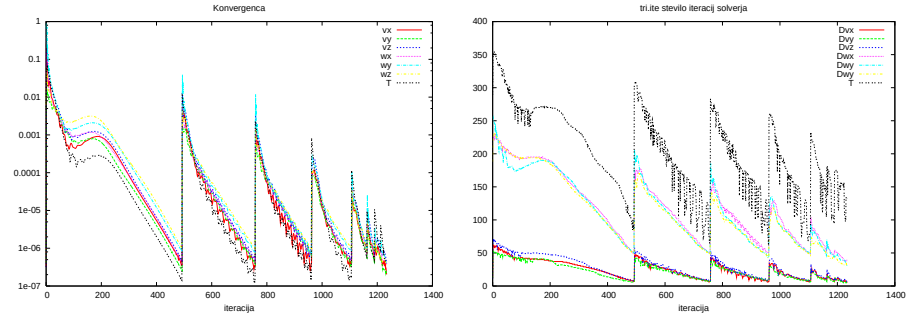
Slika 2: Primer $Ra = 10^3$. Število iteracij iterativnega solverja v odvisnosti od nelinearne iteracije. Levo zgoraj konstantna natančnost $\epsilon = 10^{-7}$, desno zgoraj $R = 100$, levo spodaj $R = 10$, desno spodaj $R = 1$.



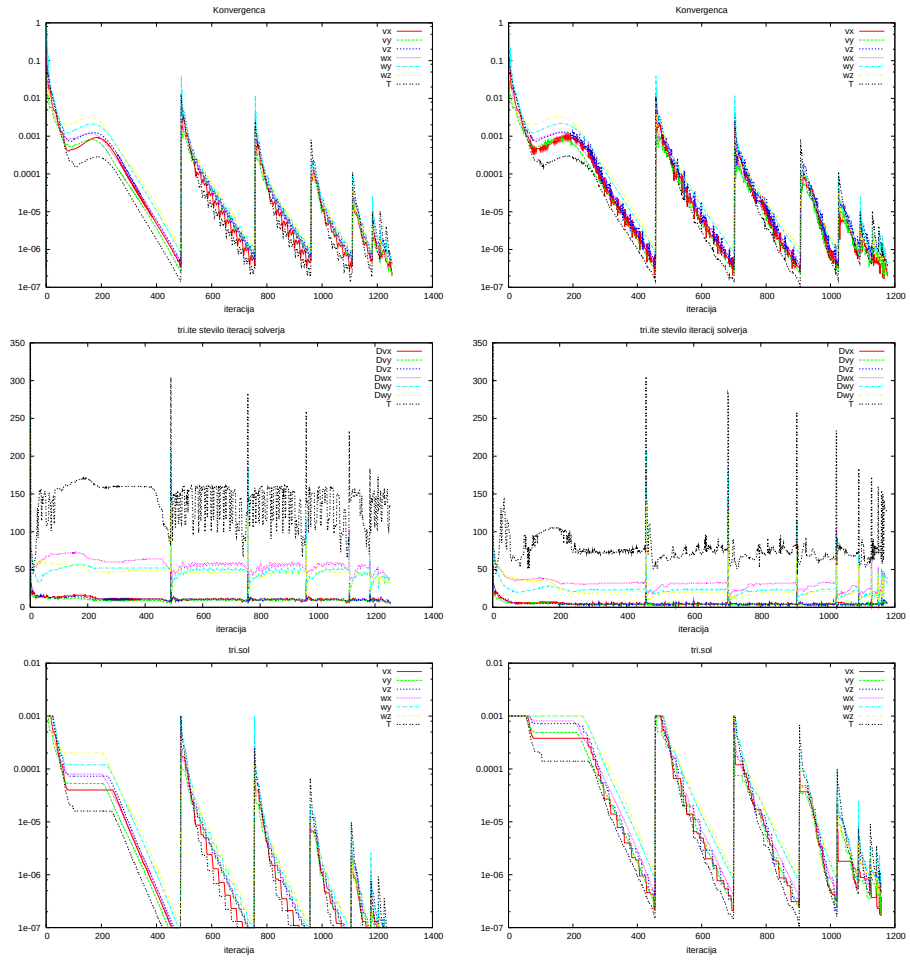
Slika 3: Primer $Ra = 10^3$. Prikaz uporabljene natančnosti solverja; Levo $R = 10$, desno $R = 1$.



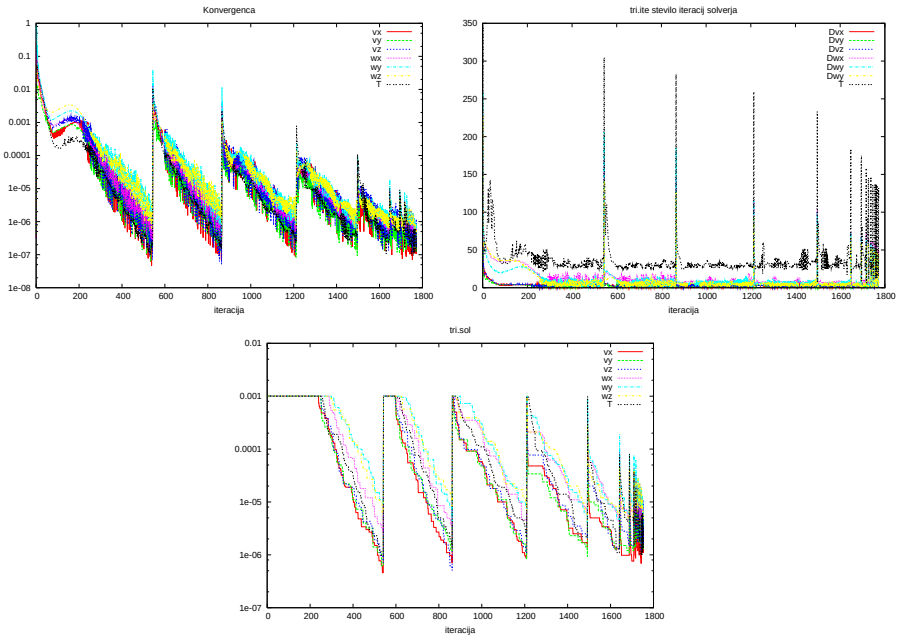
Slika 4: Primer $Ra = 10^4$. Konvergenca (zgoraj), število iteracij LSQR solverja (sredina) in zahtevana natančnost solverja (spodaj). Levo konstantna natančnost $\epsilon = 10^{-7}$, desno dinamična z $R = 10$.



Slika 5: Primer $Ra = 10^5$. Konvergenca (levo), število iteracij LSQR solverja (desno); konstantna natančnost $\epsilon = 10^{-7}$



Slika 6: Primer $Ra = 10^5$. Konvergenca (zgoraj), število iteracij LSQR solverja (sredina) in zahtevana natančnost solverja (spodaj). Levo dinamična s $R = 10$, desno $R = 1$.



Slika 7: Primer $Ra = 10^5$, $R = 0.1$. Konvergenca (levo zgoraj), število iteracij LSQR solverja (desno zgoraj) in zahtevana natančnost solverja (spodaj). Pri tako majhnem R pride do velikih oscilacij v konvergenci.

3 Zaključki

Izdelan je bil blok diagonalni predpogojevalec za predoločene sisteme enačb. Predpogojevalec je učinkovit šele, ko so bloki relativno veliki. Takrat se število iteracij iterativnega solverja zmanjša, a je čas izračuna predpogojevalca večji, kot je pridobitev časa zaradi zmanjšanega števila iteracij. Blok diagonalni predpogojevalec je tako uporaben zgolj v primerih, ko z enako sistemsko matriko rešimo sistem za veliko različnih desnih strani.

Izdelan je bil algoritem za dinamično spreminjanje natančnosti iterativnega solverja predoločenih sistemov enačb. Pokazano je bilo, da je algoritem deluje stabilno in da zmanjša število iteracij solverja na približno polovico. Za delovanje algoritem potrebuje nastavitev minimalne in maksimalne natančnosti solverja. Za maksimalno predlagamo vrednosti $\epsilon_{max} = 10^{-3}$, minimalna pa naj bo 10 krat manjša od konvergenčnega kriterija. Za primer $\epsilon_{err} = 10^{-6}$ torej vzamemo $\epsilon_{min} = 10^{-7}$. Ugotovljeno je bilo, da naj parameter R zavzema vrednosti med $1 \leq R \leq 100$. Priporočamo uporabo $R = 10$.

Literatura

[1] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8:43–71, 1982.

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes - The Art of Scientific computing, Second Edition*. Cambridge University Press, 1997.