Blokirani algoritmi za QR faktorizaciju

Kiš, Ivan

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivan Kiš

BLOKIRANI ALGORITMI ZA QR FAKTORIZACIJU

Diplomski rad

Voditelj rada:
doc. dr. sc. Nela Bosner

Zagreb, rujan 2019.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Jedna od najvažnijih faktorizacija u području numeričke linearne algebre je QR faktorizacija. Njezine najpoznatije primjene su rješavanje sustava linearnih jednadžbi, metoda najmanjih kvadrata, a nezamjenjiva je i kao dio algoritama za određivanje svojstvenih vrijednosti matrice. Kako bismo riješili neke od tih problema koristimo QR faktorizaciju pomoću Householderovih (QRHR) ili Givensovih rotacija (QRGR). Međutim, njihova primjena je prilično neefikasna jer se bazira na BLAS 1 i BLAS 2 [5] operacijama i koristi sporiju RAM memoriju. Zato je u interesu znanstvenika razviti metode koje bi preradile QRHR i QRGR tako da efikasno koriste brzu cache memoriju. U ovom radu ćemo zato obraditi blokirane verzije QR algoritama koje se oslanjaju na BLAS 3 operacije. Za algoritam baziran na Householderovim reflektorima opisat ćemo dvije verzije blokiranja: pomoću WY reprezentacije produkta reflektora [1] i pomoću VTV reprezentacije produkta koji koristi manje memorije od WY [2]. Za algoritam baziran na Givensovim rotacijama također će se opisati generiranje akumuliranih rotacija u obliku matrica malih dimenzija [3].

Poglavlje 1

QR faktorizacija, osnovni algoritmi

1.1 QR faktorizacija

Definicija 1.1.1. *Za realnu matricu $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ kažemo da ima ortogonalne stupce ako vrijedi $A^TA = I$.*

Jednostavnije rečeno, matrica ima ortogonalne stupce ako joj stupci čine ortonormiran skup, tj. skalarni produkt svaka dva različita stupca je 0, a norma svakog stupca je 1. Ako je $m = n$, onda je $A^T = A^{-1}$, pa vrijedi i

$$AA^T = AA^{-1} = I$$

Kvadratne matrice s ortogonalnim stupcima zovemo ortogonalnim matricama. Sada navedimo neke korisne činjenice o matricama s ortogonalnim stupcima.

Propozicija 1.1.2. *Neka su $A \in \mathbb{R}^{m\times n}$ i $B \in \mathbb{R}^{n\times l}$, pri čemu je $m \geq n \geq l$, matrice s ortogonalnim stupcima. Tada je i njihov produkt $AB \in \mathbb{R}^{m\times l}$ matrica s ortogonalnim stupcima.*

Dokaz. Iz definicije izlazi

$$(AB)^T(AB) = B^TA^TAB = B^TI_nB = B^TB = I_l.$$

□

Propozicija 1.1.3. *Neka je $A \in \mathbb{R}^{m\times n}$, $m \geq n$, matrica s ortogonalnim stupcima i $x \in \mathbb{R}^n$ proizvoljan vektor. Tada vrijedi*

$$||Ax||_2 = ||x||_2$$

pri čemu $||\cdot||_2$ označava 2-normu,

$$||x||_2 = \sqrt{x^Tx}.$$

Dokaz. Iz definicije 2-norme izlazi

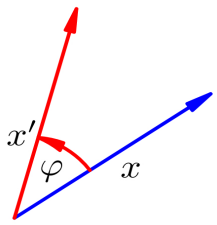
$$\|Ax\|^2 = (Ax)^T(Ax) = x^T A^T Ax = x^T Ix = x^T x = \|x\|_2^2$$

□

Definicija 1.1.4. *Matrica $A \in \mathbb{R}^2$ je matrica rotacije u ravnini ako vrijedi*

$$A(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix},$$

Ova transformacija rotira svaki vektor $x \in \mathbb{R}^2$ za kut φ, u smjeru obrnutom od kazaljke na satu.



Slika 1.1: $x' = A(\varphi)x$

Definicija 1.1.5. *Za realnu matricu $A \in \mathbb{R}^{m\times n}$ kažemo da je gornjetrokutasta (u slučaju $m \neq n$ koristi se i naziv gornjetrapezoidna) ako za sve elemente A_{ij} za koje je $i > j$ vrijedi $A_{ij} = 0$.*

Za realnu matricu $A \in \mathbb{R}^{m\times n}$ kažemo da je donjetrokutasta (u slučaju $m \neq n$ koristi se i naziv donjetrapezoidna) ako za sve elemente A_{ij} za koje je $i < j$ vrijedi $A_{ij} = 0$.

Sada navodimo bez dokaza još neke tvrdnje koje vrijede za gornjetrokutaste i donjetrokutaste matrice.

Propozicija 1.1.6. *Produkt dviju gornjetrokutastih (donjetrokutastih) matrica je gornjetrokutasta (donjetrokutasta) matrica.*

Propozicija 1.1.7. *Gornjetrokutasta (donjetrokutasta) kvadratna matrica je regularna ako i samo ako su joj svi elementi na dijagonali različiti od 0. Tada je njezin inverz gornjetrokutasta (donjetrokutasta) matrica.*

Definicija 1.1.8. *Neka je $A \in \mathbb{R}^{m\times n}$ matrica gdje je $m \geq n$. QR faktorizacija ili QR dekompozicija matrice A je rastav*

$$A = QR,$$

gdje je $Q \in \mathbb{R}^{m\times l}$ matrica s ortogonalnim stupcima, a $R \in \mathbb{R}^{l\times n}$ gornjetrokutasta matrica. Ako je $l = n$ tada govorimo o skraćenoj QR faktorizaciji, a ako je $l = m$ tada govorimo o punoj QR faktorizaciji.

1.2 Gram-Schmidtov postupak ortogonalizacije

Teorem 1.2.1. *Neka je $A \in \mathbb{R}^{m\times n}$ i rang$(A) = n$. Tada postoji jedinstvena faktorizacija oblika*

$$A = QR,$$

gdje je $Q \in \mathbb{R}^{m\times n}$ matrica s ortogonalnim stupcima, a $R \in \mathbb{R}^{n\times n}$ gornjetrokutasta s pozitivnim elementima na dijagonali.

Dokaz. Tvrdnju dokazujemo Gram-Schmidtovim postupkom ortogonalizacije (algoritam 1).

Algoritam dokazuje postojanje tražene faktorizacije, ali pažljivim promatranjem vidimo da su svi elementi matrica Q i R, zapravo, jedini mogući izbor, čime dobivamo jedinstvenost. □

Algoritam 1 QR faktorizacija Gram-Schmidtovim postupkom ortogonalizacije

Require: $A \in \mathbb{R}^{m\times n}$

A=[a_1 a_2 $\cdots$ a_n] ($a_1, a_2, \ldots, a_n$ su stupci matrice A)
for i=1:n **do**
    $b_i \leftarrow a_i - \sum_{j=1}^{n-1} \langle a_i, q_j\rangle q_j$
    $q_i \leftarrow \frac{b_i}{\|b_i\|_2}$ (Uvjet rang(A)=n osigurava $\|b_i\|_2 \neq 0$)
    for j=1:i-1 **do**
        $r_{ji} \leftarrow \langle q_j, a_i\rangle$
    end for
    $r_{ii} \leftarrow \|b_i\|_2$
end for
Q=[q_1 q_2 $\cdots$ q_n] ($q_1, q_2, \ldots, q_n$ su stupci matrice Q)

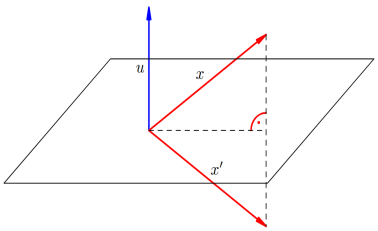
$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}$$

Gram-Schmidtov postupak koristan je za razvoj teorije, no u praksi se rijetko koristi za QR faktorizaciju zbog numeričke nestabilnosti. Naime, često je dobivena matrica Q daleko od ortogonalne. Najčešća dva algoritma koja se koriste u praksi su bazirana na Givensovim rotacijama i Householderovim reflektorima.

1.3 Householderovi reflektori

Definicija 1.3.1. *Neka je $u \in \mathbb{R}^n$ vektor 2-norme 1. Householderov reflektor je linearni operator definiran sa $H_u := I - 2uu^T$.*

Naziv "reflektor" dolazi iz prostorne predodžbe. Naime, operator H, zapravo, reflektira vektor u s obzirom na hiperravninu okomitu na vektor u. Također, vektor u nazivamo Householderov vektor.



Slika 1.2: $x' = Hx$

Propozicija 1.3.2. *Householderovi reflektori su simetrične i ortogonalne matrice.*

Dokaz. Simetričnost vrijedi jer je:

$$H_u^T = (I - 2uu^T)^T = I^T - 2(uu^T)^T = I - 2uu^T = H_u.$$

Ortogonalnost vrijedi jer je:

$$\begin{aligned} H_u H_u^T = H_u H_u &= (I - 2uu^T)(I - 2uu^T) = I - 2uu^T - 2uu^T + 4uu^T uu^T \\ &= I - 4uu^T + 4u\|u\|_2^2 u^T = I - 4uu^T + 4uu^T = I \end{aligned}$$

□

Za QR faktorizaciju treba konstruirati Householderov reflektor koji poništava sve elemente vektora osim prvog, tj. takav da za vektor $x \in \mathbb{R}^n$ vrijedi

$$Hx = \begin{bmatrix} c \\ 0 \\ \vdots \\ 0 \end{bmatrix} = ce_1.$$

Zbog Propozicije 1.1.3 odmah slijedi da je

$$\|x\|_2 = \|Hx\|_2 = \|ce_1\|_2 = |c| \cdot \|e_1\|_2 = |c|$$

Raspišemo li tu jednadžbu imamo

$$Hx = (I - 2uu^T)x = x - 2u(u^T x) = ce_1$$

Premještanjem pribrojnika dobivamo

$$u = \frac{1}{2(u^T x)}(x - ce_1)$$

pa je u linearna kombinacija x i e_1. Preciznije, mora biti

$$u = \alpha(x - ce_1),$$

za neki broj α. Nadalje, u mora biti jedinične norme, a paralelan s vektorom $x \pm \|x\|_2 e_1$ pa prema tome slijedi

$$u = \frac{x \pm \|x\|_2 e_1}{\|x \pm \|x\|_2 e_1\|_2}$$

Zbog numeričke nestabilnosti, uobičajeno se uzima predznak kojeg ima prva komponenta vektora x, koju označavamo sa x_1, tako da ne dođe do katastrofalnog kraćenja. Taj izbor se zapisuje kao

$$u = \frac{x + \text{sign}(x_1)\|x\|_2 e_1}{\|x + \text{sign}(x_1)\|x\|_2 e_1\|_2}.$$

Bez obzira na izbor predznaka, vrijedi

$$\begin{aligned} H_v x &= x - 2\frac{(x \pm \|x\|_2 e_1)(x \pm \|x\|_2 e_1)^T}{(x \pm \|x\|_2 e_1)^T(x \pm \|x\|_2 e_1)}x = x - 2\frac{(x \pm \|x\|_2 e_1)(x \pm \|x\|_2 e_1)^T x}{x^T x \pm \|x\|_2 x_1 \pm \|x\|_2 x_1 + x^T x} \\ &= x - 2\frac{(x \pm \|x\|_2 e_1)(x^T x \pm \|x\|_2 x_1)}{2(x^T x \pm \|x\|_2 x_1)} = x - (x \pm \|x\|_2 e_1) = \mp\|x\|_2 e_1 \end{aligned}$$

Samo provođenje QR faktorizacije reflektorima je vrlo jednostavno. Prvo se prvi stupac svede samo na prvi element, a nakon toga se postupak provodi na podmatrici (bez prvog retka i stupca) transformirane matrice. Postupak se ponavlja dok se ne iscrpe svi stupci. Uzmimo za primjer matricu dimenzija 5×5 i ilustrirajmo generalnu ideju algoritma. Pretpostavimo da smo već u prva dva stupca matrice A dobili željeni oblik, tada će matrica A izgledati ovako:

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \otimes & \times & \times \\ 0 & 0 & \otimes & \times & \times \\ 0 & 0 & \otimes & \times & \times \end{bmatrix}$$

Sada je potrebno odrediti Householderovu matricu koja će djelovati na naznačeni vektor na sljedeći način:

$$\overline{H_3}\begin{bmatrix}\otimes\\ \otimes\\ \otimes\end{bmatrix} = \begin{bmatrix}\times\\ 0\\ 0\end{bmatrix}$$

Zbog dimenzija matrice $\overline{H_3}$ ne možemo s njim djelovati na matricu A, pa ju "proširujemo" kao što je prikazano:

$$H_3 = diag(I, \overline{H}_3) = \begin{bmatrix} I & 0 \\ 0 & \overline{H_3}\end{bmatrix}$$

i dobivamo

$$H_3H_2H_1A = \begin{bmatrix}\times & \times & \times & \times & \times\\ 0 & \times & \times & \times & \times\\ 0 & 0 & \times & \times & \times\\ 0 & 0 & 0 & \times & \times\\ 0 & 0 & 0 & \times & \times\end{bmatrix}$$

Ovaj postupak ponavljamo dok ne dobijemo gornjetrokutastu matricu koja odgovara matrici R, a umnožak Householderovih vektora odgovara matrici Q, tj. $H_4H_3H_2H_1A = R$ pa stavljajući $Q = H_1H_2H_3H_4$ slijedi $A = QR$. Primjetimo da se jednom dobivene nule u vodećim stupcima neće pokvariti primjenom sljedećih reflektora. Dani algoritam sada možemo i formalno zapisati:

Algoritam 2 QR faktorizacija korištenjem Householderovih reflektora

Require: $A \in \mathbb{R}^{m\times n}$

1: **for** i=1:n **do**
2: $x \leftarrow A_{i:m,i}$
3: $v \leftarrow \frac{x \pm \|x\|_2 e_1}{\|x \pm \|x\|_2 e_1\|_2}$
4: $H_i \leftarrow \begin{bmatrix} I_{i-1} & 0 \\ 0 & I - 2uu^T\end{bmatrix}$
5: $A \leftarrow H_iA$
6: **end for**
7: $R = A$
8: $Q \leftarrow H_1H_2 \dots H_n$ Q je ortogonalna po propoziciji 1.1.2)

Važno je primjetiti da se djelovanje Householderovim reflektorom ne računa eksplicitnim kreiranjem matrice operatora H, već se na jedan vektor primjenjuje ovako:

$$Hx = x - 2u(u^Tx),$$

odnosno na matricu A se primjenjuje ovako:

$$HA = A - 2u(u^T A).$$

Dakle, djelovanje Householderovog reflektora na vektor je operacija linearne vremenske složenosti (svodi se na operacije između 2 vektora, poput skalarnog produkta), dok je djelovanje Householderovog reflektora na matricu operacija kvadratne vremenske složenosti (svodi se na operacije između vektora i matrice, poput množenja vektora matricom). Radi se o "level 1 BLAS", odnosno "level 2 BLAS" operacijama biblioteke BLAS.

1.4 Givensove rotacije

Givensove rotacije $G \in \mathbb{R}^{2\times 2}$ su matrice rotacije koje rotiraju vektore u ravnini tako da drugu komponentu ponište ili postave na nulu, tj. ako imamo vektor $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in \mathbb{R}^2$ matrica G će na njega djelovati na sljedeći način:

$$Gx = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ 0 \end{bmatrix}$$

Iz sustava određujemo koliki su $\sin\varphi$ i $\cos\varphi$:

$$x_1 \cos\varphi - x_2 \sin\varphi = y_1, \tag{1.1}$$
$$x_1 \sin\varphi + x_2 \cos\varphi = 0 \tag{1.2}$$

Iz (1.2) slijedi da je u slučaju kada je $x_2 = 0$ matrica G jednaka jediničnoj matrici I, inače imamo da je

$$x_2 \cos\varphi = -x_1 \sin\varphi. \tag{1.3}$$

Iskoristimo li $\sin^2\phi + \cos^2\phi = 1$ kvadriranjem dobivamo:

$$x_2^2 \cos^2\varphi = x_1^2 - x_1^2 \cos^2\varphi$$
$$x_1^2 = (x_1^2 + x_2^2)\cos^2\varphi$$

te za $x_1^2 + x_2^2 \neq 0$, tj. $\|x\| \neq 0$ vrijedi

$$\cos^2\varphi = \frac{x_1^2}{x_1^2 + x_2^2}, \text{ tj. } \cos\varphi = \frac{\pm x_1}{\sqrt{x_1^2 + x_2^2}}.$$

Ukoliko želimo da y_1 bude pozitivan, uzmemo da je

$$\cos\varphi = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}$$

a tada iz (1.3) slijedi

$$\sin\varphi = \frac{-x_2}{\sqrt{x_1^2 + x_2^2}}, \text{ tj. } y_1 = \frac{x_1^2 + x_2^2}{\sqrt{x_1^2 + x_2^2}}$$

i dobivamo vektor $y = \begin{bmatrix} y_1 & 0 \end{bmatrix}^T$. Jednako tako i za matrice iz $\mathbb{R}^{n\times n}$ možemo konstruirati Givensove rotacije. To bi bile matrice oblika

$$G(i,k,\varphi) = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & \text{i} & & \text{k} & & \end{array} \\ \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\varphi & \cdots & -\sin\varphi & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \sin\varphi & \cdots & \cos\varphi & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \end{array} \begin{array}{c} \\ \\ \text{i} \\ \\ \text{k} \\ \\ \\ \end{array}$$

pri čemu se podmatrica $\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$ nalazi na presjeku i-tog i k-tog retka odnosno stupca. Ostali dijagonalni elementi su jednaki 1, a preostali elementi izvan dijagonale su jednaki 0.

Vrijednosti $\sin\varphi$ i $\cos\varphi$ računaju se na prethodno opisan način, tj. za $x \in \mathbb{R}^n$

$$\cos\varphi = \frac{x_i}{\sqrt{x_i^2 + x_k^2}} \tag{1.4}$$

$$\sin\varphi = \frac{-x_k}{\sqrt{x_i^2 + x_k^2}} \tag{1.5}$$

Tada djelovanjem Givensove rotacije $G \in \mathbb{R}^{n\times n}$ na vektor x dobivamo vektor $y \in \mathbb{R}^n$ oblika:

$$y_j = \begin{cases} x_j, & \text{za } j \neq i, j \neq k \\ 0, & \text{za } j = k \\ \frac{x_i^2 + x_k^2}{\sqrt{x_i^2 + x_k^2}}, & \text{za } j = i \end{cases}$$

odnosno poništavamo k-tu komponentu vektora x, i-tu mijenjamo, dok ostale komponente ostaju nepromijenjene. U praksi postoje i puno efikasniji načini računanja $\sin\varphi$ i $\cos\varphi$ od (1.4) i (1.5). Način računanja sinusa i kosinusa prezentiran u Algoritmu 3 je numerički stabilniji, jer se brine da ne dođe do overloada pod korijenom.

Algoritam 3 Izračunavanje kuta rotacije

Require: x_i, x_j

1: **if** $x_j = 0$ **then**
2: $\quad \cos\varphi = 1, \sin\varphi = 0$
3: **else if** $|x_j| > |x_i|$ **then**
4: $\quad \tau = \frac{-x_i}{x_j}, \sin\varphi = \frac{1}{\sqrt{1+\tau^2}}, \cos\varphi = \tau\sin\varphi$
5: **else** $\tau = \frac{-x_j}{x_i}, \cos\varphi = \frac{1}{\sqrt{1+\tau^2}}, \sin\varphi = \tau\cos\varphi$
6: **end if**
7: $x_j = 0$

Također, vrlo često se za izračunavanje kuta rotacije koristi i "level 1 BLAS" rutina *_rotg*.

Pokažimo sada kako odrediti QR faktorizaciju korištenjem Givensovih rotacija. Neka je $A \in \mathbb{R}^{m\times n}$, gdje je $m \geq n$. Pomoću Givensovih rotacija odredimo QR rastav matrice, odnosno odredimo ortogonalnu matricu $Q \in \mathbb{R}^{m\times m}$ i gornjetrokutastu $R \in \mathbb{R}^{m\times n}$ tako da vrijedi $A = QR$. Uzmimo za primjer matricu A dimenzije 4×3 i ilustrirajmo generalnu ideju algoritma:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(1,2)}$$

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{(3,4)} R$$

Kako Givensove rotacije poništavaju samo jedan element u matrici koju množimo, mi ćemo s $G_l(i,k,\varphi) \in R^{m\times m}$ označiti matricu koja poništava element a_{kl} (element u k-tom retku i l-tom stupcu), a kut φ izračunava za zadane a_{il} i a_{kl}. Poništavanje vršimo po sljedećem algoritmu:

U praksi ćemo umjesto matrice rotacija koristiti njenu podmatricu koja sadrži samo vrijednosti $\sin\varphi$ i $\cos\varphi$ te s njom djelovati samo na odgovarajuće retke matrice A. Kako bismo to izvršili, koristimo "level 1 BLAS" rutinu *_rot*.

Algoritam 4 QR faktorizacija korištenjem Givensovih rotacija

Require: $A \in \mathbb{R}^{m\times n}$

1: $R = A, Q = I$
2: **for** j=1:n **do**
3: **for** i=m:-1:j+1 **do**
4: Izračunaj kut φ koristeći algoritam 3 (tj. elemente Givensove rotacije $G_j(i-1, i, \varphi)$)
5: $R = G_j(i-1, i, \varphi)R$ (poništava element a_{ij})
6: $Q = QG_j(i-1, i, \varphi)$
7: **end for**
8: **end for**
9: $R \in \mathbb{R}^{m\times n}$ gornjetrokutasta, $Q \in \mathbb{R}^{m\times m}$ ortogonalna

1.5 Rješavanje problema najmanjih kvadrata pomoću QR faktorizacije

Za kraj uvodnog dijela, predstavljamo jedan primjer iz primjene koji se rješava korištenjem QR faktorizacije. Rješenje problema najmanjih kvadrata je jedinstveno u slučaju kada matrica sustava A ima linearno nezavisne stupce, te se može riješiti QR faktorizacijom uz svojstvo ortogonalne matrice Q da je $\|Qx\| = \|x\|$. Neka je $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, te neka je $\text{rang}(A) = n$ i neka je $A = QR$ rastav matrice A. Tada vrijedi:

$$\|Ax - b\|^2 = \|QRx - QQ^T b\|^2 = \|Q(Rx - Q^T b)\|^2 = \|Rx - Q^T b\|^2.$$

Matricu R možemo zapisati kao

$$R = \begin{bmatrix} R_0 \\ 0 \end{bmatrix},$$

gdje je R_0 gornjetrokutasta regularna kvadratna matrica reda n, a vektor $Q^T b$ zapišemo kao

$$Q^T b = \begin{bmatrix} c \\ d \end{bmatrix},$$

pri čemu je $c \in \mathbb{R}^n$, a $d \in \mathbb{R}^{m-n}$. Imamo

$$\|Ax - b\|^2 = \|Rx - Q^T b\|^2 = \left\| \begin{bmatrix} R_0 x - c \\ -d \end{bmatrix} \right\|^2 = \|R_0 x - c\|^2 + \|d\|^2.$$

Sada trokutasti sustav $R_0 x = c$ ima jedinstveno rješenje x za koje je $\|R_0 x - c\| = 0$.

Budući da d ne ovisi o x, vrijednost $\|Ax-b\|$ se ne može više smanjiti pa je x jedinstveno rješenje problema najmanjih kvadrata.

Poglavlje 2

Blokirani algoritmi

U Algoritmu 2 smo pokazali kako korištenjem BLAS 1 i BLAS 2 rutina možemo odrediti QR faktorizaciju matrice A. U pravilu se veća efikasnost postiže povećanom upotrebom BLAS 3 rutina poput matričnog množenja jer su one optimizirane tako da bolje iskorištavaju cache memoriju računala. Zato je bolje koristiti algoritam koji veću količinu posla prebacuje na BLAS 3 funkcije.

2.1 WY reprezentacija

U poglavlju 1.3 pokazali smo kako izgleda Householderov reflektor H i kako pomoću njega možemo stupac po stupac poništavati elemente ispod dijagonale te uzastopnim ponavljanjem doći do matrice R.

Prisjetimo se da smo pri dobivanju QR faktorizacije trebali primjenjivati produkte Householderovih matrica. Problem je što produkti Householderovih matrica nisu Householderove matrice. Ipak, postoji način kako efikasno implementirati produkt Householderovih matrica[1]. To je tzv. WY reprezentacija.

Pretpostavimo da moramo izračunati produkt točno r Householderovih matrica

$$H_i = I - 2u_i u_i^T, \qquad \|u_i\|_2 = 1$$

tako da dobijemo

$$Q_r = H_1 \cdots H_r$$

Tvrdimo da se Q_r može zapisati u obliku

$$Q_r = I + W_r Y_r^T,\ W_r, Y_r \in \mathbb{R}^{n \times r} \tag{2.1}$$

gdje je Y_r donja trapezoidna matrica.

Definiramo rekurziju

$$W_i = [\ W_{i-1}, -2Q_{i-1}u_i]\ ,\ Y_i = [\ Y_{i-1}, u_i] \tag{2.2}$$

pri čemu je

$$W_1 = -2u_1,\ Y_1 = u_1$$

Tvrdnju dokazujemo indukcijom po r. Ako je $r = 1$, onda je

$$Q_1 = I + W_1 Y_1^T = I - 2u_1 u_1^T = H_1$$

Korak indukcije. Pretpostavimo da se Q_r može zapisati kao

$$Q_r = I + W_r Y_r^T.$$

Tada vrijedi

$$Q_{r+1} = Q_r H_{r+1} = Q_r(I - 2u_{r+1}u_{r+1}^T) = Q_r - 2Q_r u_{r+1}u_{r+1}^T \tag{2.3}$$

$$= I + W_r Y_r^T - 2Q_r u_{r+1}u_{r+1}^T. \tag{2.4}$$

S druge strane, iz definicije rekurzije (2.2) slijedi

$$W_{r+1}Y_{r+1}^T = [\ W_r, -2Q_r u_{r+1}]\ \cdot \begin{bmatrix} Y_r^T \\ u_{r+1}^T \end{bmatrix} = W_r Y_r^T - 2Q_r u_{r+1}u_{r+1}^T. \tag{2.5}$$

Uvrstimo li (2.5) u (2.4) dobivamo

$$Q_{r+1} = I + W_{r+1}Y_{r+1}^T$$

što smo i htjeli pokazati.

Dani rezultat sada možemo i formalno iskazati:

Lema 2.1.1. *Neka je $Q = I + WY^T$ $n \times n$ ortogonalna matrica, i neka su $W, Y \in \mathbb{R}^{n\times j}$. Ako je $H = I - 2uu^T$, $u \in \mathbb{R}^n$, $\|u\|_2 = 1$ i $z = -2Qu$, onda je*

$$Q_+ = QH = I + W_+ Y_+^T$$

gdje su $W_+ = [\ W\ z]$ i $Y_+ = [\ Y\ u]$ $n \times (j+1)$ matrice.

Dokaz.

$$QH = (I + WY^T)(I - 2uu^T) = I + WY^T - 2Quu^T$$
$$= I + WY^T + zu^T = I + \Big[W\ z\Big]\Big[Y\ u\Big]^T$$

□

Sada možemo konstruirati algoritam koji generira matrice W i Y iz danih vektora u_j koji generiraju reflektore:

Algoritam 5 Generiranje W i Y matrica

```
1: for j=1:r do
2:     if j=1 then
3:         w ← [ −2u_1 ] ; Y ← [ u_1 ]
4:     else
5:         z ← −2(I + WY^T)u_j; W ← [ W z] ; Y ← [ Y u_j]
6:     end if
7: end for
```

Primjetimo da je Y matrica Householderovih vektora i da je donjetrokutasta. Očito nam je glavni zadatak u generiranju WY reprezentacije (2.1) računanje matrice W.

Algoritam 2 je bogat operacijama matrično-vektorskog množenja(BLAS 2). Korištenjem blokirane Householderove reprezentacije, Algoritam 6 će biti bogat operacijama matrično-matričnog množenja(BLAS 3).

Prije samog algoritma, na jednostavnom primjeru ćemo ilustrirati ideju. Pretpostavimo da je $n = 9$ i da je "blokirani parametar" $b = 3$. Prvi korak je generiranje Householderovih reflektora H_1, H_2 i H_3 kao u algoritmu 2. Međutim, za razliku od algoritma 2, ovdje H_1, H_2 i H_3 primjenimo na podmatricu $A(:, 1 : 3)$. Nakon toga generiramo blok reprezentaciju $H_1H_2H_3 = I + W_1Y_1^T$ te izvedemo level-3 ažuriranje

$$A(:, 4 : 9) = (I + WY^T)^T A(:, 4 : 9)$$

Nadalje, generiramo H_4, H_5 i H_6 kao u algoritmu 2. Ove transformacije ne primjenjujemo na $A(:, 7 : 9)$ sve dok ne pronađemo njihovu blok reprezentaciju $H_4H_5H_6 = I + W_2Y_2^T$.

Struktura Householderovih vektora koji definiraju matrice $H_{i+1}, \ldots, H_j$ implicira da je prvih i redaka u W_k i Y_k jednako 0. Ova činjenica će se pokazati u praktičnoj primjeni algoritma. Pravilan način particioniranja u Algoritmu 6 je

$$A = [\, A_1, \ldots, A_N] \;\;, N = \lceil n/b \rceil$$

gdje je blok stupac A_k procesuiran u k-tom koraku koji izgleda ovako:

$$H_{((k-1)b)} \cdots H_1 A = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & \overline{A}_k & B \end{bmatrix}$$

pri čemu je $m - (k - 1)b$ broj redaka matrice $\overline{A}_k$ i $n - kb$ broj stupaca matrice B, dok b označava širinu blok matrice. Izračunamo QR faktorizaciju od $\overline{A}_k$ i ortogonalnu matricu (u WY obliku) primjenimo na B.

Algoritam 6 Blokirani WY

Require: $A \in \mathbb{R}^{m \times n}$

Require: $b \in \mathbb{N}$ je veličina bloka

1: $k = 0$
2: **for** $i = 0, b, 2b$: broj blokova $\cdot\, b$ **do**
3: $j = \min(i + b, n)$;
4: $k = k + 1$
5: Koristeći Algoritam 2 gornjetrokutariziraj $A(i + 1 : m, i + 1 : j)$ generirajući Householderove matrice $H_{i+1}, \ldots, H_j$
6: Iskoristi Algoritam 5 kako bi dobio blok reprezentaciju $I + W_k Y_k^T = H_{i+1} : H_j$
7: $A(i + 1 : m, j + 1 : n) = (I + W_k Y_k^T)^T A(i + 1 : m, j + 1 : n)$
8: **end for**
9: $R = A$
10: $Q = (I + W_1 Y_1^T) \cdots (I + W_k Y_k^T)$, Q je ortogonalna prema propoziciji 1.1.2

2.2 Kompaktna WY reprezentacija

U ovom dijelu ćemo pokazati kako možemo modificirati WY reprezentaciju. WY reprezentacija zahtjeva korištenje puno memorije ako su spremljeni svi blokirani Householderovi faktori pa je korisno naći rastav koji zahtjeva manje prostora, tj. za spremanje W i Y nam trebaju dva polja dimenzija $m \times j$. Schreiber i van Loan[2] su pokazali da se produkt $Q = I + WY^T$ može zapisati i kao

$$Q = I + YTY^T \tag{2.6}$$

gdje je $Y \in \mathbb{R}^{m\times j}$ donjetrokutasta i $T \in \mathbb{R}^{j\times j}$ gornjetrokutasta pa zajedno stanu u jedno polje dimenzija $m \times j$. Zapis (2.6) nazivamo kompaktna WY reprezentacija matrice Q.

Teorem 2.2.1 (Kompaktno WY ažuriranje)**.** *Neka je $Q = I + YTY^T \in \mathbb{R}^{m\times m}$ ortogonalna matrica i neka je $Y \in \mathbb{R}^{m\times j}$, $m > j$ i $T \in \mathbb{R}^{j\times j}$ gornjetrokutasta. Neka je $H = I - 2uu^T$ Householderova matrica, pri čemu je $u \in \mathbb{R}^m$ i $\|u\|_2 = 1$. Ako označimo*

$$Q_+ = QH,$$

onda je

$$Q_+ = I + Y_+T_+Y_+^T$$

gdje je

$$Y_+ = [\ Y\ u\ ] \in \mathbb{R}^{m\times(j+1)}$$

i

$$T_+ = \begin{bmatrix} T & z \\ 0 & p \end{bmatrix}$$

gdje je $\rho = -2$ i $z = -2TV^Tu$.

Dokaz. Izračunajmo tražene veličine

$$\begin{aligned} I + Y_+T_+Y_+^T &= I + [\ Y\ u\ ] \begin{bmatrix} T & z \\ 0 & \rho \end{bmatrix} \begin{bmatrix} Y^T \\ u^T \end{bmatrix} \\ &= I + [\ Y\ u\ ] \begin{bmatrix} TY^T + zu^T \\ \rho u^T \end{bmatrix} \\ &= I + YTY^T + Yzu^T + \rho uu^T \end{aligned}$$

To je jednako

$$Q_+ = QH = (I + YTY^T)(I - 2uu^T) = I + YTY^T - 2YTY^Tuu^T - 2uu^T$$

kad je $\rho = -2$ i $z = -2TY^Tu$ □

Algoritam 7 Generiranje Y i T matrica

Require: $u_1, u_2, \ldots, u_r$ vektori r Householderovih reflektora na $\mathbb{R}^m$

1: $Y \leftarrow [\, u_1]$
2: $T \leftarrow [\, -2]$
3: **for** j=2:r **do**
4: $\quad z \leftarrow -2TY^T u_j$
5: $\quad Y \leftarrow [\, Y\ u_j]$
6: $\quad T \leftarrow \begin{bmatrix} T & z \\ 0 & -2 \end{bmatrix}$
7: **end for**

Sada možemo konstruirati algoritam koji generira matrice Y i T iz danih vektora u_j koji generiraju reflektore:

Neka je matrica $Y \in \mathbb{R}^{m\times j}$ donjetrokutasta, a $T \in \mathbb{R}^{j\times j}$. Takve operatore nazivamo blok-reflektorima. Time dobivamo algoritam u kojem je veća količina posla prebačena na "level 3 BLAS" rutinu *_dgemm* što popriličnо ubrzava algoritam.

Algoritam 8 Blokirani VTV

Require: $A \in \mathbb{R}^{m\times n}$
Require: $b \in \mathbb{N}$ je veličina bloka

1: $k = 0$
2: **for** $i = 0, b, 2b$: broj blokova $\cdot\, b$ **do**
3: $\quad j = \min(i + b, n)$;
4: $\quad k = k + 1$
5: $\quad$ Koristeći Algoritam 2 gornjetrokutariziraj $A(i + 1 : m, i + 1 : j)$ generirajući Householderove matrice $H_{i+1}, \ldots, H_j$
6: $\quad$ Iskoristi Algoritam 7 kako bi dobio blok reprezentaciju $I + Y_k T_k Y_k^T = H_{i+1} : H_j$
7: $\quad A(i + 1 : m, j + 1 : n) = (I + Y_k T_k Y_k^T)^T A(i + 1 : m, j + 1 : n)$
8: **end for**
9: $R = A$
10: $Q = (I + Y_1 T_1 Y_1^T) \cdots (I + Y_k T_k Y_k^T)$, $\quad Q$ je ortogonalna prema propoziciji 1.1.2

2.3 Blokirane Givensove rotacije

U poglavlju 1.4 pokazali smo kako izgleda matrica rotacije G i kako pomoću nje možemo element po element poništavati elemente ispod dijagonale te uzastopnim ponavljanjem doći do matrice R.

Naš je cilj pronaći način kako preraspodjeliti i grupirati rotacije u blokirane skupove kako bismo dozvolili efikasnu upotrebu BLAS 3 rutina u ažuriranju matrice A.

Kako bismo opisali osnovnu ideju grupiranja, za početak uzmimo da je $G^{(j)}_{i-1,i}$ Givensova rotacija definirana u ravnini određenoj koordinatama $i-1$ i i. $G^{(j)}_{i-1,i}$ poništava element u i-tom retku i j-tom stupcu matrice A. Sljedeća slika ilustrira niz Givensovih rotacija korištenih u Algoritmu 4 koje poništavaju prvih $n_b = 4$ stupaca matrice dimenzija 12×12.

$$
\begin{aligned}
&G^{(1)}_{11,12} \to G^{(1)}_{10,11} \to G^{(1)}_{9,10} \mapsto G^{(1)}_{8,9} \to G^{(1)}_{7,8} \to G^{(1)}_{6,7} \to G^{(1)}_{5,6} \mapsto G^{(1)}_{4,5} \to G^{(1)}_{3,4} \to G^{(1)}_{2,3} \to G^{(1)}_{1,2} \mapsto \\
&G^{(2)}_{11,12} \to G^{(2)}_{10,11} \mapsto G^{(2)}_{9,10} \to G^{(2)}_{8,9} \to G^{(2)}_{7,8} \to G^{(2)}_{6,7} \mapsto G^{(2)}_{5,6} \to G^{(2)}_{4,5} \to G^{(2)}_{3,4} \to G^{(2)}_{2,3} \mapsto \\
&G^{(3)}_{11,12} \mapsto G^{(3)}_{10,11} \to G^{(3)}_{9,10} \to G^{(3)}_{8,9} \to G^{(3)}_{7,8} \mapsto G^{(3)}_{6,7} \to G^{(3)}_{5,6} \to G^{(3)}_{4,5} \to G^{(3)}_{3,4} \mapsto \\
&G^{(4)}_{11,12} \to G^{(4)}_{10,11} \to G^{(4)}_{9,10} \to G^{(4)}_{8,9} \mapsto G^{(4)}_{7,8} \to G^{(4)}_{6,7} \to G^{(4)}_{5,6} \to G^{(4)}_{4,5}
\end{aligned}
$$

Slika 2.1: Niz Givensovih rotacija korištenih za poništavanje prva 4 stupca matrice A

U [6] je pokazano da rotacije $G^{(j_1)}_{i_1-1,i_1}$ i $G^{(j_2)}_{i_2-1,i_2}$ komutiraju, ako vrijedi $i_2 > i_1 - 1$. Tada iz Slike 2.1 slijedi da rotacije $G^{(2)}_{11,12}, G^{(2)}_{10,11}$ i $G^{(3)}_{11,12}$ komutiraju s čitavim nizom $G^{(1)}_{1,2} \to G^{(1)}_{2,3} \to \cdots \to G^{(1)}_{8,9}$ pa slijedi da rotacije možemo grupirati na sljedeći način:

$$
\begin{aligned}
&G^{(1)}_{11,12} \to G^{(1)}_{10,11} \to G^{(1)}_{9,10} \to G^{(2)}_{11,12} \to G^{(2)}_{10,11} \to G^{(3)}_{11,12} \to \\
&G^{(1)}_{8,9} \to G^{(1)}_{7,8} \to G^{(1)}_{6,7} \to G^{(1)}_{5,6} \to G^{(2)}_{9,10} \to G^{(2)}_{8,9} \to G^{(2)}_{7,8} \to G^{(2)}_{6,7} \to \\
&G^{(3)}_{10,11} \to G^{(3)}_{9,10} \to G^{(3)}_{8,9} \to G^{(3)}_{7,8} \to G^{(4)}_{11,12} \to G^{(4)}_{10,11} \to G^{(4)}_{9,10} \to G^{(4)}_{8,9} \to \\
&G^{(1)}_{4,5} \to G^{(1)}_{3,4} \to G^{(1)}_{2,3} \to G^{(1)}_{1,2} \to G^{(2)}_{5,6} \to G^{(2)}_{4,5} \to G^{(2)}_{3,4} \to G^{(2)}_{2,3} \to \\
&G^{(3)}_{6,7} \to G^{(3)}_{5,6} \to G^{(3)}_{4,5} \to G^{(3)}_{3,4} \to G^{(4)}_{7,8} \to G^{(4)}_{6,7} \to G^{(4)}_{5,6} \to G^{(4)}_{4,5}
\end{aligned}
$$

Slika 2.2: Pregrupirane rotacije

Prednost pregrupiranja rotacija je u tome što na svaki pregrupirani skup možemo učinkovito primjeniti matrično-matrično množenje. Na primjer, produkt svih rotacija u drugom skupu

iz Slike 2.2, $G_{8,9}^{(1)} \to G_{7,8}^{(1)} \to \cdots \to G_{8,9}^{(4)}$ je oblika $\begin{bmatrix} I_4 & 0 \\ 0 & U \end{bmatrix}$, pri čemu je

$$U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \tag{2.7}$$

takva da su $U_{21} \in \mathbb{R}^{4\times4}$ gornjetrokutasta i $U_{12} \in \mathbb{R}^{4\times4}$ donjetrokutasta matrica. Za općeniti blok širine n_b, matrica U je dimenzija $2n_b \times 2n_b$. Izuzetak je prvi pregrupirani skup (Slika 2.2), za koji će matrica U bti manjih dimenzija i U_{21} će biti puna matrica. Dakle, kako bismo iskoristili BLAS 3 rutinu u ažuriranju matrice A, prvo gomilamo rotacije iz pojedinog skupa u ortogonalnu matricu U koja je odgovarajućih dimenzija i onda U množimo sa odgovarajućom podmatricom od A. Prije samog algoritma ilustrirajmo ideju:

X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
0	X	X	X	X
0	0	X	X	X

X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
0	X	X	X	X
0	0	X	X	X
0	0	0	X	X
0	0	0	X	X

X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
0	X	X	X	X
0	0	X	X	X
0	0	0	X	X
0	0	0	X	X
0	0	0	X	X
0	0	0	X	X
0	0	0	X	X

Tablica 2.1: Poništavamo elemente i akumuliramo rotacije te nakon toga ažuriramo matricu slijeva

Algoritam 9 Blokirane Givensove rotacije

Require: $A \in \mathbb{R}^{m\times n}$, $Q \in \mathbb{R}^{m\times m}$, $Q = I$
Require: $b \in \mathbb{N}$ širina bloka
$l = m/b$ broj rotacijskih skupova u blok stupcu
$L = l$ prikazuje broj rotacijskih skupova u pojedinom stupcu
$ost = m\%b$ potrebno za dobivanje dimenzije prve grupe
$p = (L-1)b + 1$ početni redak podmatrice u blok stupcu
$p_{pom} = p$
$P = m$ zadnji redak podmatrice u blok stupcu
$r = P - p + 1$ broj redaka podmatrice u blok stupcu
for $i = 0, b, 2b$: broj blokova $\cdot\, b$ **do**
    $j = min(i + b, n)$
    $Q_{blok} = I$
    while $l > 0$ **do**
        Koristeći Algoritam 4 gornjetrokutariziraj $A(p : P, i+1 : j)$ i pritom akumuliraj rotacije u matricu $U \in \mathbb{R}^{r\times r}$, pri čemu je U oblika 2.7
        $A(p : P, j + 1 : n) = U(1 : r, 1 : r)A(p : P, j + 1 : n)$
        $Q_{blok} = \begin{bmatrix} I_{p-1} & 0 & 0 \\ 0 & U & 0 \\ 0 & 0 & I_{m-P} \end{bmatrix} Q_{blok}$
        $l = l - 1$, $p = p - b$
        if $r = b + ost$ **then**
            $P = P - ost$
            $r = 2b$
        else
            $P = P - b$
        end if
    end while
    $Q = \begin{bmatrix} I_i & 0 \\ 0 & Q_{blok} \end{bmatrix} Q$ ažuriranje ortogonalne matrice Q
    $L = L - 1, l = L$ prelazimo u idući stupac pa se smanjio broj rotacijskih skupova
    $P = m$
    $p = p_{pom}$
    $r = P - p + 1$
end for

Poglavlje 3

Usporedba algoritama

Algoritmi su implementirani na računalu Intel Pentium N3250 sa 4 procesorske jezgre i 2 MB L2 cache memorije. Testiranja će pokazati uspješnost korištenja memorijske hijerarhije. Testirat ćemo vrijeme izvršavanja algoritama ovisno o dimenzijama matrice. Korištene su različite slučajne matrice. Pokazalo se da su za trajanje izvršavanja algoritma bitne samo dimenzije matrice. Pritom distribucija elemenata tih matrica nema značajan utjecaj. Stoga ćemo prikazati vremena izvršavanja dobivena samo korištenjem matrica s jednoznamenkastim pozitivnim elementima.

3.1 Algoritmi bazirani na Householderovim reflektorima

Za širinu bloka podmatrica odabrali smo $b = 25$ koji se pokazao dobrim pri testiranju na kvadratnim matricama. HolderQR predstavlja rezultate algoritma 2, WY algoritma 6 te YTY algoritma 8.

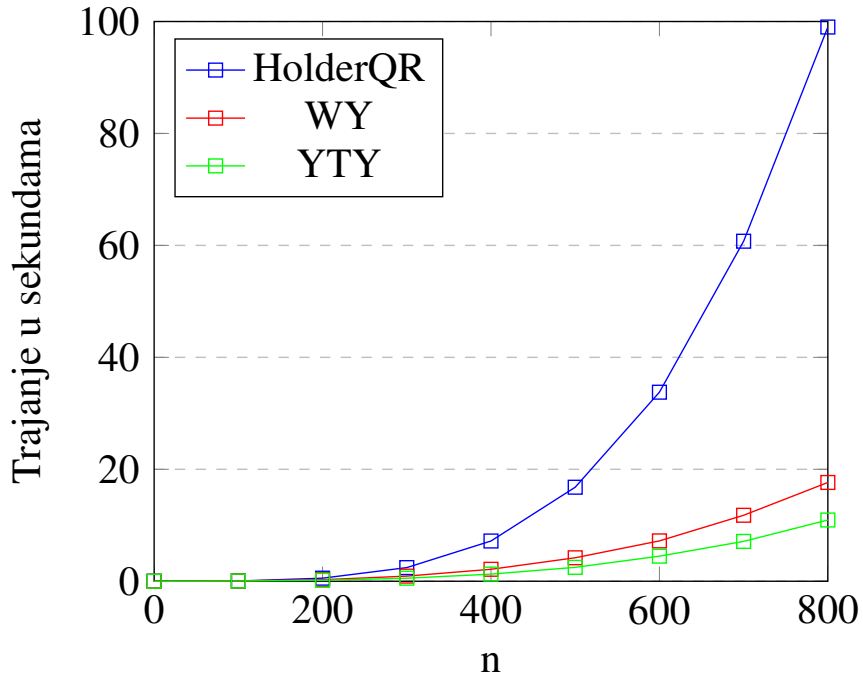
Na sljedećim tablicama vidimo ovisnost vremena izvršavanja o dimenzijama matrice. Programi su izvršeni korištenjem 1 procesorske jezgre. U tablici 3.1 vidimo ponašanje algoritma za kvadratne matrice, dok je na slici 3.1 grafički prikaz vremena izvršavanja.

Tablica 3.1:

dimenzije	HolderQR	WY	YTY
100×100	0.044282	0.056129	0.041045
200×200	0.547145	0.272813	0.173478
300×300	2.436637	0.915731	0.539172
400×400	7.187694	2.136226	1.272350
500×500	16.798055	4.193439	2.482246
600×600	33.773360	7.209351	4.470402
700×700	60.746644	11.790303	7.119844
800×800	99.009872	17.646779	10.939837

Brzina izvršavanja algoritama (u sekundama) na matricama dimenzija $n \times n$

Slika 3.1:

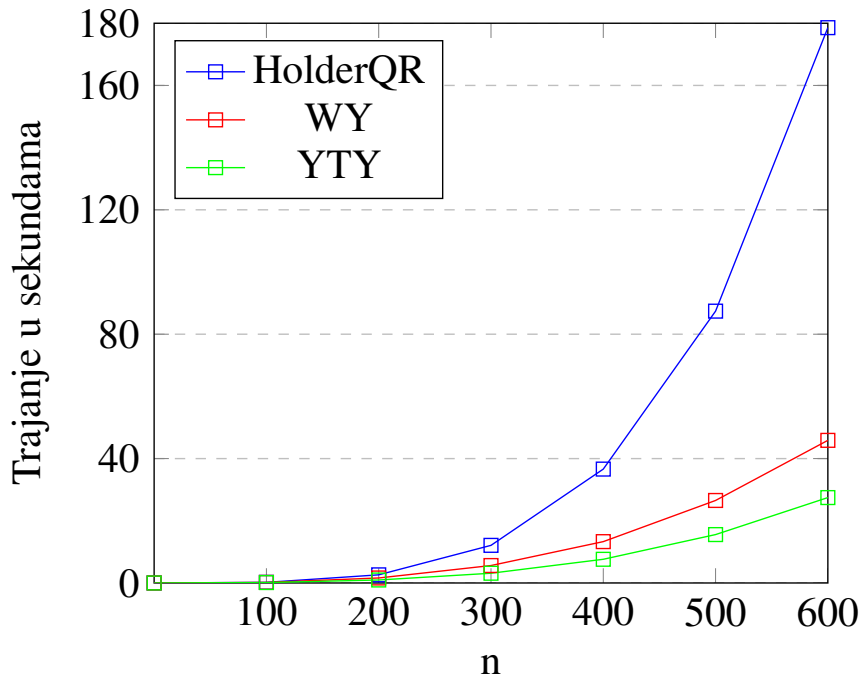


Tablica 3.2:

dimenzije	HolderQR	WY	YTY
200×100	0.200818	0.183826	0.138888
400×200	2.618716	1.608196	0.961241
600×300	12.100019	5.609546	3.131202
800×400	36.622363	13.297335	7.621255
1000×500	87.415377	26.534846	15.565882
1200×600	178.576481	45.866103	27.462608

Brzina izvršavanja algoritama (u sekundama) na matricama dimenzija $2n \times n$

Slika 3.2:



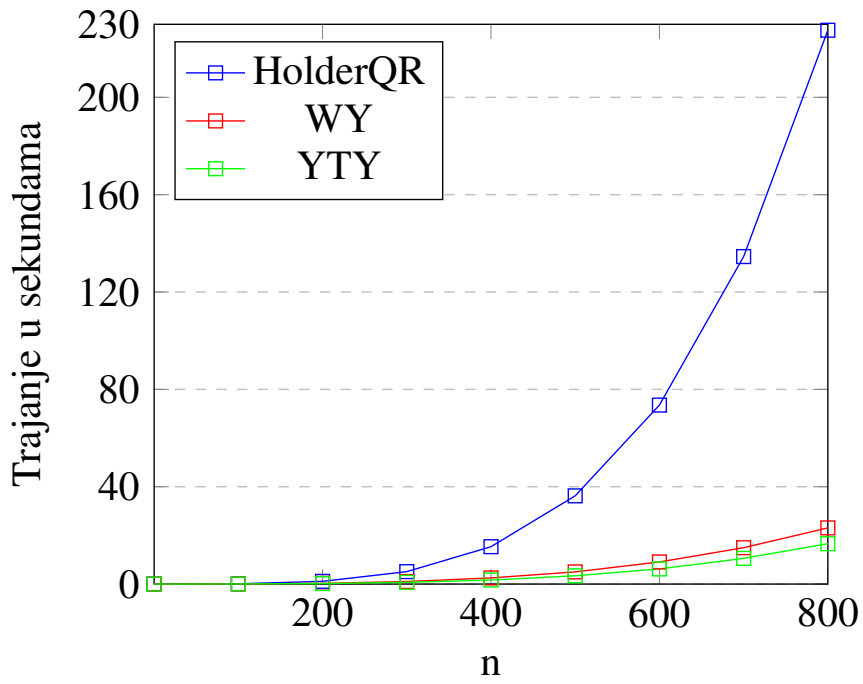
U tablici 3.2(grafički prikaz je na slici 3.2) su algoritmi testirani na matricama dimenzija $2n \times n$.

Tablica 3.3:

dimenzije	HolderQR	WY	YTY
100×200	0.109145	0.050058	0.041500
200×400	1.115721	0.297117	0.200341
300×600	5.099709	1.043456	0.703710
400×800	15.348803	2.516528	1.645243
500×1000	36.245021	4.998113	3.395774
600×1200	73.542414	9.086742	6.219780
700×1400	134.564906	14.939563	10.548035
800×1600	227.561033	23.074434	16.583166

Brzina izvršavanja algoritama (u sekundama) na matricama dimenzija $n \times 2n$

Slika 3.3:



U tablici3.3(grafički prikaz je na slici 3.3) su algoritmi testirani na matricama dimenzija $n \times 2n$.

Tablica 3.4:

dimenzije	b=10	b=15	b=25
100×100	0.029609	0.029240	0.041045
200×200	0.129304	0.136549	0.173478
300×300	0.463146	0.461436	0.539172
400×400	1.249884	1.207238	1.272350
500×500	2.711747	2.492895	2.482246
600×600	5.107057	4.488776	4.470402
700×700	8.813032	7.455853	7.119844
800×800	14.265610	11.94541	10.939837
900×900	21.832181	17.994335	16.441833
1000×1000	32.221915	25.878745	22.763843

Brzina izvršavanja algoritama (u sekundama) na matricama dimenzija $n \times n$, b predstavlja odabranu širinu bloka

Testiranja pokazuju da omjer broja redaka i stupaca u matrici uvelike utječe na vremensku efikasnost blokiranih algoritama. Također, lako je primjetiti da su blokirani algoritmi višestruko brži od neblokiranog algoritma. Budući da kompaktna WY reprezentacija zahtjeva manje prostora od obične, bilo je i očekivano da će imati bolju vremensku efikasnost.

Kako bismo pokazali da je izbor širine blok matrice također iznimno bitan, testirali smo vremensku efikasnost QR faktorizacije pomoću YTY reprezentacije na kvadratnim matricama različitih dimenzija. Iz tablice 3.4 možemo primjetiti da povećanje dimenzija matrica zahtjeva i povećanje širine bloka kako bismo dobili optimalnu vremensku efikasnost.

3.2 Algoritmi bazirani na Givensovim rotacijama

Za širine bloka podmatrica odabrali smo $b = 40$, $b = 80$, $b = 100$ i $b = 150$. GivensQR predstavlja rezultate algoritma 4, dok su u naredna četiri stupca rezultati algoritma 9. Kako bismo potvrdili da je izbor širine blok matrice iznimno bitan, testirali smo vremensku efikasnost blokiranog algoritma na kvadratnim matricama različitih dimenzija (tablica 3.5). Testiranja pokazuju da omjer broja redaka i stupaca u matrici, isto kao i u slučaju kod Householdera, utječe na vremensku efikasnost blokiranog algoritma. Također, lako je primjetiti da je blokirani algoritam višestruko vremenski efikasniji od neblokiranog algoritma.

Tablica 3.5:

dimenzije	GivensQR	b=n/2	b=n/4	b=n/8
200×200	0.291507	0.247185	0.163199	0.110580
400×400	3.131560	2.403951	1.197886	0.690322
600×600	11.395272	9.062100	4.552334	2.350638
800×800	24.511853	20.965170	12.295097	5.321261
1000×1000	47.046972	43.070911	25.909430	10.969535

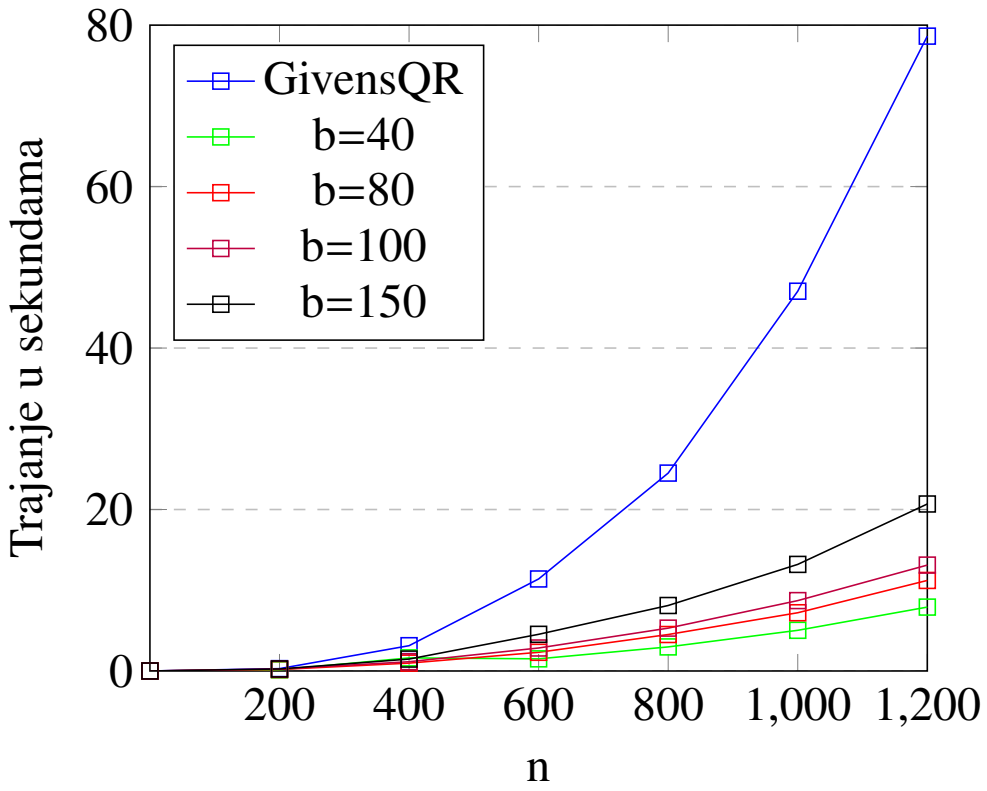
U tablici 3.6 vidimo ponašanje algoritma za kvadratne matrice, dok je na slici 3.4 grafički prikaz vremena izvršavanja. U tablici 3.7(grafički prikaz je na slici 3.5) su algoritmi testirani na matricama dimenzija $n \times 2n$. U tablici 3.8(grafički prikaz je na slici 3.6) su algoritmi testirani na matricama dimenzija $n \times 2n$. Također, lako je primjetiti da je blokirani algoritam vremenski efikasniji od neblokiranog algoritma.

Tablica 3.6:

dimenzije	GivensQR	b=40	b=80	b=100	b=150
200×200	0.291507	0.141423	0.185730	0.243520	0.247278
400×400	3.131560	0.620537	0.973155	1.146396	1.475826
600×600	11.395272	1.507317	2.313436	2.847063	4.542652
800×800	24.511853	2.981349	4.517293	5.307486	8.111414
1000×1000	47.046972	5.037659	7.213327	8.717906	13.203787
1200×1200	78.622647	7.913682	11.215459	13.116901	20.684642

Brzina izvršavanja algoritama (u sekundama) na matricama dimenzija $n \times n$

Slika 3.4:

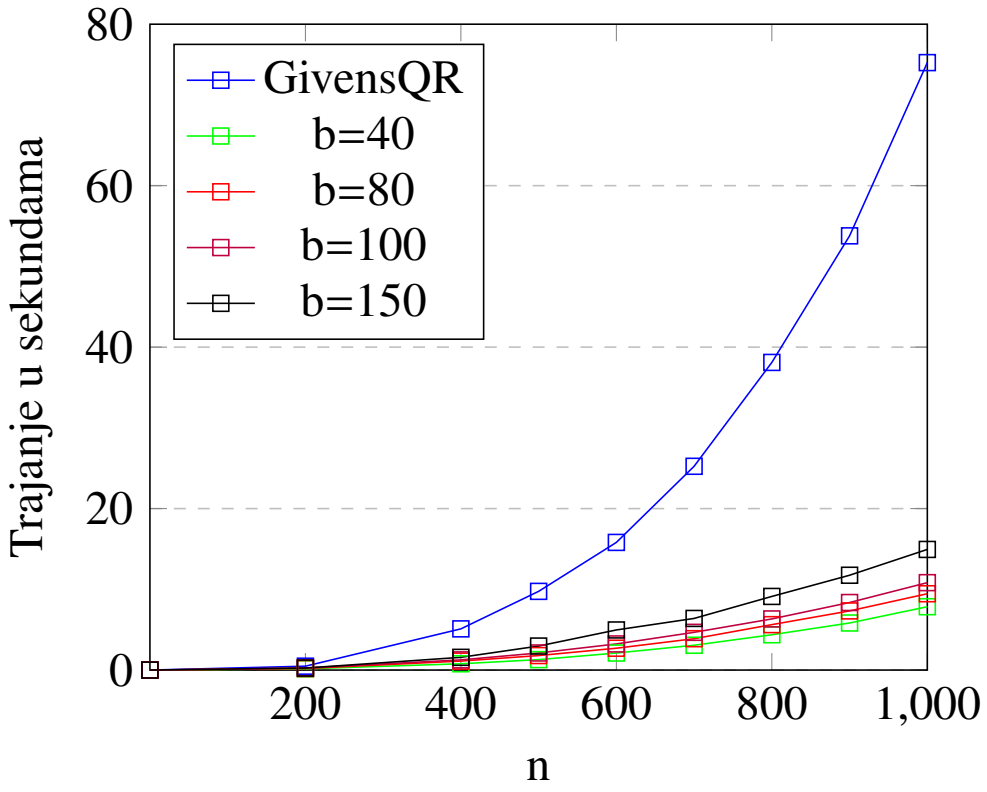


Tablica 3.7:

dimenzije	GivensQR	b=40	b=80	b=100	b=150
200×400	0.479328	0.157464	0.193605	0.261343	0.250271
400×800	5.104633	0.778937	1.111117	1.263424	1.577223
500×1000	9.744633	1.293848	1.797197	2.118018	2.974615
600×1200	15.828500	2.109543	2.706312	3.231375	4.968051
700×1400	25.234346	3.073300	3.880461	4.673921	6.395919
800×1600	38.104633	4.373848	5.625197	6.333395	9.120078
900×1800	53.758500	5.829343	7.327312	8.365669	11.945174
1000×2000	75.241346	7.845170	9.454761	10.827931	14.942375

Brzina izvršavanja algoritama (u sekundama) na matricama dimenzija $n \times 2n$

Slika 3.5:

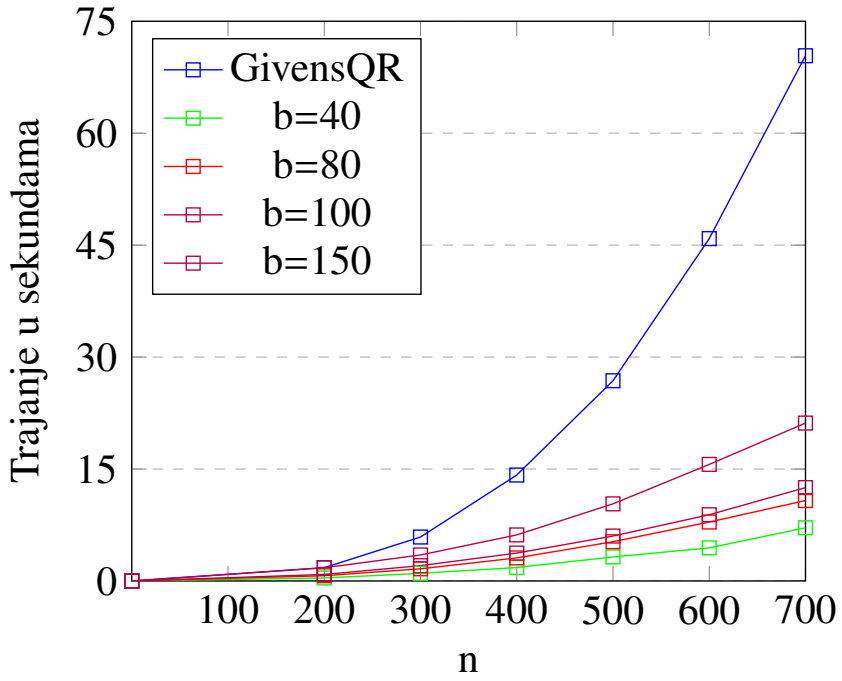


Tablica 3.8:

dimenzije	GivensQR	b=40	b=80	b=100	b=150
400×200	1.779328	0.408552	0.701598	0.873180	1.177243
600×300	5.880244	1.015652	1.628284	2.050862	3.482260
800×400	14.195633	1.801848	3.089238	3.735004	6.171903
1000×500	26.821540	3.218943	5.261189	6.018805	10.332036
1200×600	45.878446	4.434630	7.895611	8.873096	15.617967
1400×700	70.129666	7.124717	10.773518	12.509181	21.159547

Brzina izvršavanja algoritama (u sekundama) na matricama dimenzija $2n \times n$

Slika 3.6:



3.3 Točnost algoritama

Nakon što smo u prethodnim poglavljima pokazali brzinu naših algoritama, potrebno je provjeriti i njihovu točnost. Kako bismo dobili grešku u algoritmu, morali smo izračunati faktore Q i R. Tada možemo izračunati povratnu grešku ρ u obliku $\rho = \frac{\|A-QR\|_F}{\|A\|_F}$ Točnost algoritama testirat ćemo na kvadratnim matricama. U tablici (3.9) promatramo greške za algoritme bazirane na Householderovim reflektorima. Izabrali smo širinu bloka podmatrica $b = 25$, isto kao u testiranjima brzine. HolderQR predstavlja rezultate algoritma 2, WY algoritma 6 te YTY algoritma 8. Iz tablice (3.9) lako je uočiti da povećanjem dimenzija matrica raste i greška. Također, blokirani algoritmi su "točniji" od neblokiranog. U tablici (3.10) promatramo greške blokiranih algoritama koristeći različite širine blokova podmatrica. Dani rezultati nam pokazuju da greška ne mora ovisiti o zadanoj širini bloka.

Tablica 3.9:

dimenzije	HolderQR	WY	YTY
100×100	1.185471	0.088805	0.854137
200×200	1.726639	1.393948	1.115690
300×300	1.891134	1.370380	1.181183
400×400	2.232845	1.549274	1.100864
500×500	2.582880	1.786726	1.209845

Greške algoritama na matricama dimenzija $n \times n$, $\rho = c \cdot 10^{-15}$

Tablica 3.10:

WY				
n\b	10	25	50	100
200	1.323918	1.393948	1.515449	1.488646
400	1.435374	1.549275	1.560676	1.595699
600	1.737269	1.874926	1.786622	1.891358
800	1.841707	2.090578	1.828473	2.042099
VTV				
n\b	10	25	50	100
200	1.025875	1.115693	1.316507	1.419174
400	0.097736	1.100864	1.046456	1.204529
600	1.151939	1.319801	1.184779	1.395457
800	1.119584	1.414100	1.120425	1.348926

Greške algoritama na matricama dimenzija $n \times n$, $\rho = c \cdot 10^{-15}$

U tablici (3.11) promatramo greške za algoritme bazirane na Givensovim rotacijama. Izabrali smo širine bloka podmatrica $b = 60$, $b = 80$ i $b = 100$. GivensQR predstavlja rezultate algoritma 4, dok su u naredna tri stupca rezultati algoritma 9. Iz tablice (3.11) možemo zaključiti da povećanjem dimenzija matrica raste i greška.

Tablica 3.11:

dimenzije	GivensQR	b=60	b=80	b=100
200×200	1.779240	1.910258	1.934745	1.872623
400×400	2.334551	2.775651	2.878258	2.624128
600×600	3.550268	3.777002	3.896937	3.935897
800×800	5.101202	4.896952	5.453164	5.694092
1000×1000	5.023585	5.350827	5.609843	5.919843

Greške algoritama na matricama dimenzija $n \times n$, $\rho = c \cdot 10^{-15}$

Bibliografija

[1] C. Bischof, C. F. van Loan, *The WY representation for products of Householder matrices*, SIAM J. Sci. Stat. Comput., Vol 8, 1987, p. s2-s13

[2] R. Schreiber, C. F. van Loan, *A storage-efficient WY representation for products of Householder transformations*, SIAM J. Sci. Stat. Comput. Vol 10, 1989, p. 53-57

[3] B. Lang, *Using level 3 BLAS in rotation-based algorithms*, SIAM J. Sci. Comput., Vol 19, 1998, p. 626-634

[4] G. H. Golub, C. F. van Loan, *Matrix Computations*, Third edition, John Hopkins University Press, Baltimore, 1996.

[5] *BLAS(Basic Linear Algebra Subprograms)*:
`http://www.netlib.org/blas/documentation`

[6] B. Kagström, D. Kressner, E. S. Quintana-Ortí, G. Quintana-Ortí, *Blocked Algorithms for the Reduction to Hessenberg-Triangular Form Revisited*, LAPACK Working Note 198, February 1, 2008

Sažetak

Standardni način računanja QR faktorizacije je primjena niza elementarnih ortogonalnih transformacija na matricu, i to Householderovih reflektora ili Givensovih rotacija. Međutim, njihova primjena je prilično neefikasna jer se bazira na BLAS 1 i BLAS 2 operacijama. U ovom radu predstavljene su blokirane verzije QR algoritama koji se oslanjaju na BLAS 3 operacije i efikasno koriste brzu cache memoriju. To znači da se algoritam mora restruktuirati na način da se stupci organiziraju u blokove koji se najprije obrade, a tek onda se ostatak matrice ažurira korištenjem matrično-matričnog množenja. Na taj način smanjuje se komunikacija između brze cache memorije i sporije RAM memorije. Za algoritam baziran na Householderovim reflektorima opisali smo dvije verzije blokiranja: pomoću WY reprezentacije produkta reflektora, i pomoću YTY reprezentacije produkta koji koristi manje memorije od WY. Za algoritam baziran na Givensovim rotacijama smo opisali generiranje akumuliranih rotacija u obliku matrica malih dimenzija. Njihovom implementacijom i uspoređivanjem vremena izvršavanja neblokiranih i blokiranih verzija algoritma dobili smo višestruko vremensko ubrzanje, kao što smo i očekivali.

Summary

A standard way of calculating QR factorization is to apply a series of elemental orthogonal transformations to a matrix, namely Householder reflectors or Givens rotations. However, their implementation is quite inefficient as it is based on BLAS 1 and BLAS 2 operations. In this paper, block versions of QR algorithms are presented that rely on BLAS 3 operations and use fast cache efficiently. This means that the algorithm must be restructured in such a way that the columns are organized into blocks which are first processed, and only then the rest of the matrix is updated by using matrix-matrix multiplication. This reduces the communication between fast cache and slower RAM. For the Householder reflector algorithm, we describe two blocking versions: using the WY representation of the reflector product, and using the YTY representation of the product which uses less memory than WY. For a Givens-based rotation algorithm, we have described the generation of accumulated rotations in the form of small-dimensional matrices. By implementing them and comparing the execution time of unblocked and blocked versions of the algorithm, we obtained multiple time acceleration, as we expected.

Životopis

Ivan Kiš rođen je 21.6.1992. godine u Zagrebu. Nakon završetka osnovne škole, 2007. godine upisuje prirodoslovno-matematičku gimnaziju u srednjoj školi Tina Ujevića u Kutini. 2011. godine upisuje preddiplomski sveučilišni studij na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu. Diplomski sveučilišni studij Primijenjena matematika upisuje 2015. godine.