J. Math. and Its Appl.
ISSN: 1829-605X
Vol. 3, No. 1, May 2006, 19–25

Perbandingan Algoritma Golub Kahan dan QR Simetri untuk Dekomposisi Nilai Singular

Dieky Adzkiya*, E. Apriliani, Bandung A.S.
Jurusan Matematika
FMIPA ITS Surabaya
adzkiyadieky@telkom.net, {april,bandung}@matematika.its.ac.id

Abstrak

Estimasi variabel maupun parameter pada sistem berskala besar, khususnya dengan Filter Kalman dibutuhkan waktu komputasi yang lama. Dengan melakukan reduksi rank matriks kovariansi, waktu komputasi dapat dipercepat. Reduksi rank dapat dilakukan dengan Dekomposisi nilai singular (SVD), reduksi rank ini tidak mengurangi tingkat akurasi hasil estimasi.

Pada paper ini dibahas perbandingan dua algoritma untuk dekomposisi nilai singular, yaitu Golub Kahan dan QR Simetri. Dilakukan uji empiris pada berbagai macam matriks untuk membandingkan waktu kerja kedua algoritma tersebut. Dari hasil simulasi diperoleh bahwa algoritma QR Simetri memerlukan waktu komputasi yang lebih cepat dibandingkan dengan algoritma Golub Kahan.

Kata Kunci: *reduksi rang, SVD, Golub Kahan, QR Simetri.*

*Mhs S2 Jur. Mat. ITS

1. Pendahuluan

Dewasa ini banyak permasalahan yang membutuhkan perkiraan/estimasi keadaan pada masa mendatang, seperti estimasi ketinggian air sungai, estimasi penyebaran polusi udara, prakiraan cuaca dan sebagainya. Salah satu metode estimasi yang dapat digunakan adalah Filter Kalman. Untuk mengestimasi sistem berskala besar, Filter Kalman membutuhkan waktu komputasi yang lama, sehingga untuk mengurangi waktu komputasi tersebut dilakukan reduksi rank pada matriks kovariansi kesalahan estimasinya.

Reduksi rank merupakan proses untuk mengurangi rank suatu matriks. Dekomposisi nilai singular digunakan untuk reduksi rank tanpa mengubah karakteristiknya. Sehingga estimasi yang dihasilkan tetap akurat dan waktu komputasinya lebih cepat. Terdapat beberapa metode untuk dekomposisi nilai singular, antara lain metode Golub Kahan dan QR Simetri. Pada paper ini dibandingkan jumlah kerja yang dilakukan oleh kedua algoritma dan juga waktu komputasinya. Simulasi untuk membandingkan waktu komputasi digunakan Matlab.

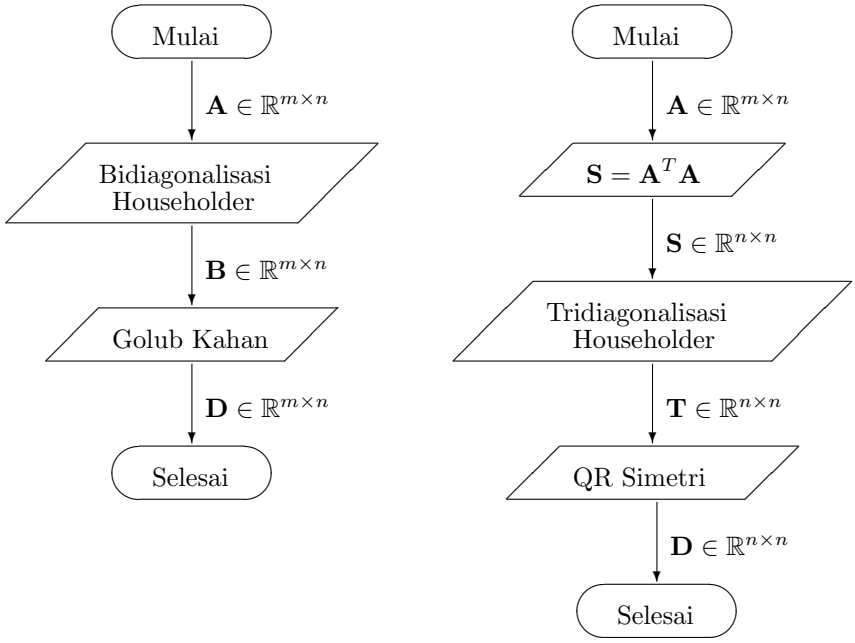
2. Dekomposisi Nilai Singular

Dekomposisi nilai singular (Singular Value Decomposition) yang selanjutnya disingkat dengan SVD merupakan suatu metode untuk melakukan reduksi rank suatu matriks dengan cara mendekomposisi matriks tersebut kedalam matriks Diagonal yang berisi nilai singular dan matriks-amtriks ortogonal dengan kolom-kolom matriks berupa vektor singular yang bersesuaian dengan nilai singular.

Pada paper ini dikaji dekomposisi nilai singular dengan metode Golub Kahan dan metode QR simetri. Dalam melakukan dekomposisi, Algoritma Golub Kahan mengubah matriks awal menjadi matriks bidiagonal kemudian dicari nilai singularnya sedangkan algoritma QR Simetri mengubah matriks awal menjadi matriks simetri, kemudian diubah menjadi matriks tridiagonal baru kemudian dicari nilai singularnya.

Proses pencarian nilai singular dari kedua metode tersebut dapat digambarkan seperti Gambar (1). Dengan $\mathbf{A}$ merupakan matriks sebarang, sedangkan $\mathbf{B}$ dan $\mathbf{T}$ masing masing menyatakan matriks bidiagonal dan tridiagonal, matriks simetri dilambangkan dengan $\mathbf{S}$ sedangkan $\mathbf{D}$ merupakan matriks diagonal.

Metode diagonalisasi matriks, baik bidiagonalisasi maupun tridiagonalisasi dilakukan dengan menggunakan refleksi Householder [1]. Tridiagonalisasi dapat dipercepat dengan memanfaatkan sifat simetri.



Gambar 1: Flowchart proses SVD menggunakan Golub Kahan dan QR Simetri

3. Algoritma Golub Kahan

Dekomposisi nilai singular menggunakan Golub Kahan membutuhkan rotasi Givens. Penjelasan yang lebih terperinci dapat dilihat di [1]. Algoritma Golub Kahan dapat dituliskan sebagai berikut.

Input : $\mathbf{A} \in \mathbb{R}^{m\times n}$; $m, n > 0$
Output : $\mathbf{B} = \mathbf{U}^T\mathbf{A}\mathbf{V} \in \mathbb{R}^{n\times n}$, matriks diagonal
`Golub_Kahan(`$\mathbf{A}$`,` $\mathbf{B}$`,` $\mathbf{U}$`,` $\mathbf{V}$`,` m`,` n`)`
lakukan bidiagonalisasi pada $\mathbf{A}$ untuk menghitung
$\mathbf{B} = (\mathbf{U}_1\mathbf{U}_2\ldots\mathbf{U}_n)^T\,\mathbf{A}\,(\mathbf{V}_1\mathbf{V}_2\ldots\mathbf{V}_{n-2})$
`while` $q \neq n$
ganti $b_{i,i+1}$ dengan nol jika $|b_{i,i+1}| \leq \epsilon(|b_{i,i}| + |b_{i+1,i+1}|)$
untuk $i = 1 : n-1$
cari nilai q terbesar dan nilai p terkecil sedemikian hingga jika

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_{11} & 0 & 0 \\ 0 & \mathbf{B}_{22} & 0 \\ 0 & 0 & \mathbf{B}_{33} \end{pmatrix}$$

dengan $\mathbf{B}_{11} \in \mathbb{R}^{p\times p}$, $\mathbf{B}_{22} \in \mathbb{R}^{(n-p-q)\times(n-p-q)}$, $\mathbf{B}_{33} \in \mathbb{R}^{q\times q}$
maka $\mathbf{B}_{33}$ diagonal dan elemen superdiagonal $\mathbf{B}_{22}$ tidak nol
if $q < n$
if terdapat nol pada diagonal $\mathbf{B}_{22}$ then
elemen superdiagonal sebaris diganti nol
else
lakukan algoritma step Golub Kahan pada $\mathbf{B}_{22}$
$\mathbf{B} = \text{diag}(\mathbf{I}_p, \mathbf{U}, \mathbf{I}_{q+m-n})^T \mathbf{B}\, \text{diag}(\mathbf{I}_p, \mathbf{V}, \mathbf{I}_q)$
end
end
end
end Golub_Kahan

Dalam menghitung kerja yang dilakukan algoritma Golub Kahan, operasi yang selevel dianggap sama [2], seperti penjumlahan dan pengurangan serta perkalian dan pembagian. Algoritma di atas melakukan $O(mn^2 - \frac{1}{3}n^3)$ penjumlahan, $O(mn^2 - \frac{1}{3}n^3)$ perkalian dan $O(n^2)$ akar dalam worst case.

Memori yang dibutuhkan untuk menyimpan matriks $\mathbf{B}$, $\mathbf{U}$ dan $\mathbf{V}$ tergantung ukuran input jadi algoritmanya tidak work in place. Tetapi dengan menyimpan matriks $\mathbf{B}$ di matriks $\mathbf{A}$ dan merepresentasikan matriks $\mathbf{U}$ dan $\mathbf{V}$ dalam bentuk vektor dan disimpan di bagian nol matriks $\mathbf{A}$ maka algoritma tersebut menjadi work in place, karena memori ekstra yang dibutuhkan untuk menyimpan p, q dan i yang tidak tergantung ukuran input. Space yang diperlukan sebanyak mn variabel float untuk menyimpan matriks $\mathbf{A}$ dan lima buah variabel integer untuk menyimpan m, n dan kebutuhan memori ekstra.

4. Algoritma QR Simetri

Rotasi givens juga diperlukan pada algoritma QR simetri. Di bawah ini adalah algoritma QR Simetri secara singkat.

Input : $\mathbf{A} \in \mathbb{R}^{m\times n}$; $m, n > 0$
Output : $\mathbf{D} \in \mathbb{R}^{n\times n}$, matriks diagonal; $\mathbf{V} \in \mathbb{R}^{n\times n}$
QR_Simetri(**A**, **D**, **V**, m, n)
$\mathbf{D} = \mathbf{A}^T\mathbf{A}$
lakukan tridiagonalisasi pada $\mathbf{D}$ untuk menghitung
$\mathbf{T} = (\mathbf{V}_1\mathbf{V}_2 \ldots \mathbf{V}_{n-2})^T \mathbf{D}\, (\mathbf{V}_1\mathbf{V}_2 \ldots \mathbf{V}_{n-2})$
while $q \neq n$
ganti $t_{i+1,i}$ dan $t_{i,i+1}$ dengan nol jika

$|t_{i+1,i}| = |t_{i,i+1}| \leq \epsilon\left(|t_{i,i}| + |t_{i+1,i+1}|\right)$ untuk $i = 1 : n-1$

cari nilai q terbesar dan nilai p terkecil sedemikian hingga jika

$$\mathbf{T} = \begin{pmatrix} \mathbf{T}_{11} & 0 & 0 \\ 0 & \mathbf{T}_{22} & 0 \\ 0 & 0 & \mathbf{T}_{33} \end{pmatrix}$$

dengan $\mathbf{T}_{11} \in \mathbb{R}^{p\times p}$, $\mathbf{T}_{22} \in \mathbb{R}^{(n-p-q)\times(n-p-q)}$, $\mathbf{T}_{33} \in \mathbb{R}^{q\times q}$

maka $\mathbf{T}_{33}$ diagonal dan $\mathbf{T}_{22}$ tidak dapat direduksi

if $q < n$

lakukan algoritma step QR Simetri pada $\mathbf{T}_{22}$

$\mathbf{T} = \operatorname{diag}(\mathbf{I}_p, \mathbf{V}, \mathbf{I}_q)^T \mathbf{T} \operatorname{diag}(\mathbf{I}_p, \mathbf{V}, \mathbf{I}_q)$

end

end

end QR_Simetri

Dengan menggunakan asumsi yang sama dengan algoritma Golub Kahan maka algoritma QR simetri melakukan kerja penjumlahan sebanyak $O(n^3)$, kerja perkalian juga $O(n^3)$ serta kerja akar $O(n^2)$ dalam worst case.

Algoritma tersebut tidak work in place tetapi dapat diubah menjadi work in place dengan menggunakan cara pada algoritma Golub Kahan. Memori ekstra dan space yang dibutuhkan sama dengan algoritma Golub Kahan.

5. Hasil Pengujian

Algoritma Golub Kahan dan QR Simetri yang telah disusun pada bagian sebelumnya diimplementasikan menggunakan Matlab dan dibandingkan waktu komputasinya. Komputer yang digunakan untuk uji coba adalah Pentium II 400 MHz dengan memori 128 MB. Ukuran matriks yang digunakan untuk uji coba paling besar berukuran 1500×1500. Elemen-elemen matriks diambil secara acak dan setiap ukuran matriks dilakukan uji coba sebanyak 31 kali.

Dari hasil simulasi seperti pada Tabel 1 dan 2 tampak bahwa waktu komputasi untuk algoritma Golub Kahan jauh lebih besar dibandingkan dengan algoritma QR simetri untuk ukuran matriks yang sama. Sedangkan dari Gambar 2, tampak bahwa untuk jumlah kolom 750, waktu komputasi Golub Kahan semakin meningkat dengan meningkatnya jumlah baris, tetapi waktu komputasi untuk QR simetri cenderung konstan.

Pada Gambar 3 menunjukkan bahwa untuk jumlah baris 750, waktu komputasi QR simetri dan Golub Kahan semakin meningkat dengan meningkatnya jumlah kolom, tetapi peningkatan waktu komputasi Golub Kahan lebih tinggi dibandingkan dengan peningkatan waktu komputasi QR simetri.

Tabel 1: Waktu komputasi Algoritma QR Simetri

	Kolom				
b		600	900	1200	1500
a	600	72.775			
r	900	74.748	236.08		
i	1200	88.497	249.18	5594.7	
s	1500	76.821	244.28	5798.4	1066.1

Tabel 2: Waktu komputasi Algoritma Golub Kahan

	Kolom				
b		600	900	1200	1500
a	600	156.675			
r	900	212.316	463.046		
i	1200	255.778	602.777	1033	
s	1500	304.498	719.435	1291.9	1886.9

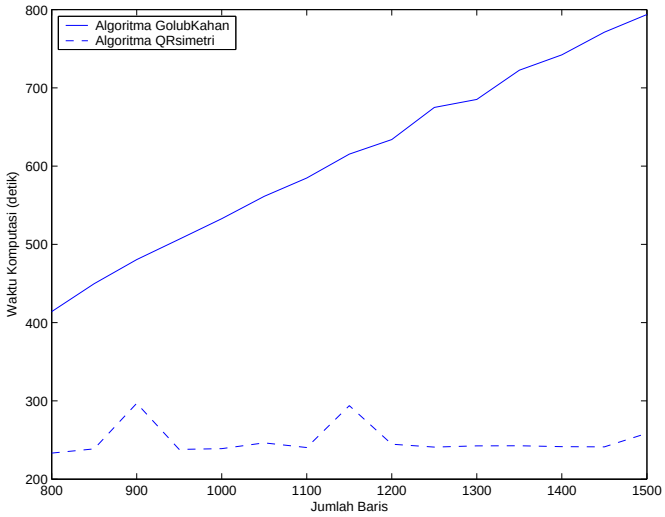
6. Kesimpulan

Beberapa hal yang dapat disimpulkan antara lain, untuk ukuran matriks yang sama, algoritma QR Simetri lebih cepat dari pada algoritma Golub Kahan. Jumlah kerja algoritma QR Simetri dan Golub Kahan berada dalam order fungsi yang sama.

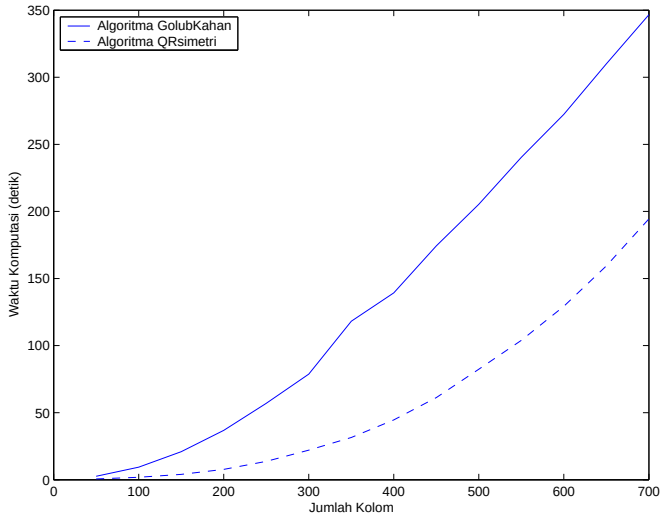
Pustaka

[1] Golub G.H. and Van Loan C.F., (1989), *Matrix Computation*, Second Edition, The John Hopkins University Press, London.

[2] Shaffer C.A., (1998), *A Practical Introduction to Data Structures and Algorithm Analysis*, Prentice Hall Inc., New Jersey.



Gambar 2: Grafik waktu komputasi algoritma Golub Kahan dan QR Simetri dengan jumlah kolom 750



Gambar 3: Grafik waktu komputasi algoritma Golub Kahan dan QR Simetri dengan jumlah baris 750