PROCEDIAMATH
The Use of Big Data for Education & Kontribusi Matematika dalam Mempertahankan Nilai Budaya dan Sastra

TRUE DAN *GHOST BREAKDOWN* PADA ALGORITMA LANCZOS TIPE A4/ORTHORES DALAM SISTEM LINIER BERDIMENSI TINGGI

Maharani [1] and Annisa Savitri[2]
Department of Mathematics, University of Jenderal
email: maharaniyusro@gmail.com [1], annisasvitri200995@gmail.com[2]

ABSTRACT

Lanczos method is an iterative method used to find the solution of a system of linear equations. Currently, the Lanczos method has been expanded and modified into several types. One type that has been considerably implemented is the Lanczos-type A_4 *(Orthores algorithm). This research examines the Orthores derivation algorithm by applying Formal Orthogonal Polynomials (FOPs). The Lanczos algorithm is very effective for solving systems of high-dimensional linear equations (SLEs). However, it is very susceptible to high iteration use or to orthogonal polynomials. This phenomenon is commonly called breakdown. Breakdown occurs because of the division by zero when the computation process takes place, and thus the algorithm halts.. In this study, the algorithm was implemented in some variety of SLEs, ranging from1000 to 10.000 dimensions.*

Keywords : *Lanczos method, formula A4/Orthores, algorithm A4/Orthores, FOPs, breakdown.*

ABSTRAK

Metode Lanczos merupakan metode iteratif yang digunakan untuk mencari solusi sistem persamaan linier. Saat ini, metode Lanczos telah dikembangkan dan dimodifikasi menjadi beberapa jenis. Salah satu jenis yang telah banyak diimplementasikan adalah metode Lanczos tipe A_4 (algoritma Orthores). Penelitian ini mengkaji tentang derivasi algoritma Orthores dengan mengaplikasikan *Formal Orthogonal Polynomials* (FOPs). Algoritma Lanczos sangat efektif untuk menyelesaikan sistem persamaan linier berdimensi tinggi. Akan tetapi dalam implementasinya sangat rentan terhadap penggunaan iterasi tinggi atau terhadap polinomial ortogonal. Fenomena ini biasa disebut *breakdown*. *Breakdown* terjadi karena adanya pembagian dengan nol saat proses komputasi berlangsung dan menyebabkan algoritma berhenti. Dalam penelitian ini algoritma akan disimulasikan pada dimensi tertentu.

Kata kunci : metode Lanczos, formula A4/Orthores, algoritma A4/Orthores, FOPs, *breakdown*.

1. PENDAHULUAN

Secara umum, penyelesaian sistem persamaan linier (SPL) terdiri dari dua metode, yaitu metode langsung dan metode iteratif. Metode langsung terdiri dari eliminasi Gauss, eliminasi Gauss-Jordan, dan aturan Cramer. Kelemahan metode langsung yaitu metode ini tidak efisien apabila digunakan untuk menyelesaikan SPL berdimensi tinggi. Sebaliknya, metode iteratif sangat efisien untuk menyelesaikan SPL berdimensi tinggi. Metode iteratif terbagi menjadi dua kelas besar, yaitu metode stasioner dan metode tak stasioner. Perbedaan antara keduanya, yaitu metode stasioner sangat sederhana dalam memformulasikan solusi aproksimasi, namun sangat lambat dalam mencapai kekonvergenan. Sebaliknya metode tak stasioner sangat cepat mencapai solusi aproksimasi yang bagus (Barret dkk,1994:5-6). Metode yang termasuk dalam metode stasioner adalah metode Jacobi, metode Gauss-seidle, *Successive Overrelaxation Method* (SOR) dan Chebisev. Metode yang termasuk dalam metode tak stasioner adalah metode subruang Krylov, metode Lanczos, metode Arnoldi dan *Conjugate Gradient Method* (C G). Pada studi ini, akan dibahas metode Lanczos untuk menyelesaikan SPL berdimensi tinggi.

Pada tahun 1950, Cornelius Lanczos menemukan metode untuk menyelesaikan masalah nilai eigen dengan cara mentransformasi suatu matriks menjadi matriks tridiagonal yang serupa (Lanczos, 1950). Tahun 1952, Lanczos mengembangkan metode tersebut untuk menyelesaikan SPL dengan mengkombinasikan dengan metode subruang Krylov. Selanjutnya, metode tersebut dikenal sebagai metode Lanczos (Lanczos, 1952). Pengembangan metode Lanczos selanjutnya dilakukan oleh Brezinski dengan menggunakan *Formal Orthogonal Polynomials* (FOPs) (Brezinski dan Sadok, 1993).

Sejumlah ilmuwan telah mengembangkan dan

memodifikasi algoritma Lanczos, hingga kini terdapat beberapa tipe. Salah satunya adalah Baheux, yang menemukan jenis A_j, B_j dan kombinasi dari $A_j \text{ dan } B_j$, untuk $j = 1, 2, \ldots 10$. Beberapa algoritma yang telah diimplementasikan oleh Baheux adalah A_4 (Orthores), A_8/B_6 (Orthodir) dan A_8/B_{10} (Orthomin) (Baheux, 1995). Selain itu, Farooq mengembangkan jenis A_j, B_j dan kombinasi dari $A_j \text{ dan } B_j$, untuk $j = 11, 12, \ldots 20$ (Farooq, 2011).

Algoritma Lanczos sangat efektif untuk menyelesaikan SPL berdimensi tinggi, namun algoritma ini sangat rentan terhadap penggunaan iterasi tinggi atau terhadap perhitungan polinomial ortogonal. Kelemahan ini yang biasa disebut dengan *breakdown*. *Breakdown* pada algoritma Lanczos merupakan fenomena yang tidak bisa dihindari, yang menyebabkan algoritma tersebut berhenti sebelum mendapatkan solusi aproksimasi yang bagus (Brezinski dkk., 1992). *Breakdown* pada algoritma Lanczos terdiri dari dua macam, yaitu *true breakdown* dan *ghost breakdown*. *True breakdown* terjadi ketika polinomial ortogonal tidak terdefinisi, hal ini disebabkan oleh pembagian dengan nol pada koefisien polinomial. Sedangkan *ghost breakdown* terjadi bukan karena polinomial ortogonal tidak terdefinisi tetapi terjadi pada relasi rekursif yang digunakan dalam perhitungan (Brezinski dan Zaglia, 1994). Berdasarkan uraian di atas, penulis tertarik untuk mengkaji penurunan algoritma Lanczos tipe A_4 (algoritma Orthores) yang di dalamnya termuat metode subruang Krylov. Lebih lanjut, akan dideteksi terjadinya *breakdown* dalam algoritma Orthores.

2. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah studi pustaka, yaitu dengan mempelajari beberapa buku dan jurnal yang berkaitan dengan penelitian ini. Adapun tahapan yang dilakukan adalah sebagai berikut:

1. Menderivasikan algoritma Orthores.

2. Menyelidiki terjadinya *breakdown* pada algoritma Orthores.

3. HASIL DAN PEMBAHASAN

3.1 Derivasi Algoritma Orthores

Menurut (Baheux, 1995), terdapat dua relasi rekursif yang digunakan dalam metode tipe Lanczos, yaitu P_k dan $P_k^{(1)}$. Relasi rekursif yang digunakan dalam formula A_4 (algoritma Orthores) adalah P_{k-2} dan P_{k-1}, dengan derajat pada kedua polinomial $k-1$ dan $k-2$. Misal didefinisikan polinomial ortogonal yang mengikuti formula A_4 sebagai

$$P_k(x) = a_k\left[(x+b_k)P_{k-1}(x)+\left(d_k x^2+e_k x+f_k\right)P_{k-2}(x)\right], \tag{1}$$

dengan $P_k(0)=1$. Apabila kondisi normal $P_k(0)=1$ diaplikasikan pada persamaan (1), maka didapatkan

$$a_k = \frac{1}{(b_k+f_k)}. \tag{2}$$

Selanjutnya, jika kedua ruas pada persamaan (1) dikalikan x^i dari kiri dan menurut (Brezinski dan Zaglia, 1994), maka persamaan (1) dapat ditulis menjadi

$$\begin{aligned} c\left(x^i P_k(x)\right) = {} & a_k c\left(x^{i+1}P_{k-1}(x)\right)+a_k b_k c\left(x^i P_{k-1}(x)\right)+a_k d_k c\left(x^{i+2}P_{k-2}(x)\right)+ \\ & a_k e_k c\left(x^{i+1}P_{k-2}(x)\right)+a_k f_k c\left(x^i P_{k-2}(x)\right), \end{aligned} \tag{3}$$

untuk $i=0,1,2,\ldots,k-1$. Jika $i=k-4$, maka persamaan (3) menjadi

$$\begin{aligned} c\left(x^{k-4}P_k(x)\right) = {} & a_k c\left(x^{k-3}P_{k-1}(x)\right)+a_k b_k c\left(x^{k-4}P_{k-1}(x)\right)+ \\ & a_k d_k c\left(x^{k-2}P_{k-2}(x)\right)+a_k e_k c\left(x^{k-3}P_{k-2}(x)\right)+ \\ & a_k f_k c\left(x^{k-4}P_{k-2}(x)\right), \\ 0 = {} & a_k d_k c\left(x^{k-2}P_{k-2}(x)\right), \\ 0 = {} & d_k. \end{aligned} \tag{4}$$

Untuk $i=k-3$, maka persamaan (3) menjadi

$$\begin{aligned} c\left(x^{k-3}P_k\left(x\right)\right) &= a_k c\left(x^{k-2}P_{k-1}\left(x\right)\right) + a_k b_k c\left(x^{k-3}P_{k-1}\left(x\right)\right) + \\ &\quad a_k d_k c\left(x^{k-1}P_{k-2}\left(x\right)\right) + a_k e_k c\left(x^{k-2}P_{k-2}\left(x\right)\right) + \\ &\quad a_k f_k c\left(x^{k-3}P_{k-2}\left(x\right)\right), \\ 0 &= a_k e_k c\left(x^{k-2}P_{k-2}\left(x\right)\right), \\ 0 &= e_k, \end{aligned} \tag{5}$$

dimana $A_k \neq 0$. Selanjutnya untuk $i = k-2$, persamaan (3) menjadi

$$\begin{aligned} c\left(x^{k-2}P_k\left(x\right)\right) &= a_k c\left(x^{k-1}P_{k-1}\left(x\right)\right) + a_k b_k c\left(x^{k-2}P_{k-1}\left(x\right)\right) + \\ &\quad a_k d_k c\left(x^{k}P_{k\text{-}2}\left(x\right)\right) + a_k e_k c\left(x^{k\text{-}1}P_{k\text{-}2}\left(x\right)\right) + \\ &\quad a_k f_k c\left(x^{k-2}P_{k-2}\left(x\right)\right), \\ f_k &= -\frac{c\left(x^{k-1}P_{k-1}\left(x\right)\right)}{c\left(x^{k-2}P_{k-2}\left(x\right)\right)}. \end{aligned} \tag{6}$$

Jika $i = k-1$ disubstitusikan ke persamaan (3) diperoleh

$$\begin{aligned} c\left(x^{k-1}P_k\left(x\right)\right) &= a_k c\left(x^{k}P_{k-1}\left(x\right)\right) + a_k b_k c\left(x^{k-1}P_{k-1}\left(x\right)\right) + \\ &\quad a_k d_k c\left(x^{k+1}P_{k-2}\left(x\right)\right) + a_k e_k c\left(x^{k}P_{k-2}\left(x\right)\right) + \\ &\quad a_k f_k c\left(x^{k-1}P_{k-2}\left(x\right)\right), \\ b_k &= -\frac{c\left(x^{k}P_{k-1}\left(x\right)\right) + f_k c\left(x^{k-1}P_{k-2}\left(x\right)\right)}{c\left(x^{k-1}P_{k-1}\left(x\right)\right)}. \end{aligned} \tag{7}$$

Selanjutnya, apabila persamaan (2), (4), (5), (6), dan (7) disubstitusi ke persamaan (1), maka diperoleh

$$P_k\left(x\right) = a_k\left[\left(x + b_k\right)P_{k-1}\left(x\right) + f_k P_{k-2}\left(x\right)\right]. \tag{8}$$

Pada kondisi ortogonal, masing-masing f_{k+1} dan b_{k+1} dapat ditulis sebagai

$$\begin{aligned} f_{k+1} &= -\frac{c\left(t^k P_k\right)}{c\left(t^{k-1}P_{k-1}\right)}, \\ \Leftrightarrow\ f_{k+1} &= -\frac{\left\langle y_k, r_k\right\rangle}{\left\langle y_{k-1}, r_{k-1}\right\rangle}, \end{aligned} \tag{9}$$

$$b_{k+1} = -\frac{\left(c\left(t^{k+1}P_k\right) - f_{k+1}c\left(t^k P_{k-1}\right)\right)}{c\left(t^k P_k\right)},$$
$$\Leftrightarrow b_{k+1} = -\frac{\langle y_k, Wr_k\rangle - f_{k+1}\langle y_k, r_{k-1}\rangle}{\langle y_k, r_k\rangle}. \quad (10)$$

Lebih lanjut, akan dicari vektor residu dan solusi pendekatan dengan mensubstitusikan matriks A pada persamaan (8) untuk menggantikan variabel x sehingga dapat ditulis

$$P_k(A) = a_k\left[(A+b_k)P_{k-1}(A) + f_k P_{k-2}(A)\right]. \quad (11)$$

Misal vektor residu didefinisikan sebagai $\mathbf{r}_{k+1} = P_{k+1}(A)\mathbf{r}_0$. Apabila persamaan (11) disubstitusikan ke $\mathbf{r}_{k+1}$, maka diperoleh

$$\begin{aligned}\mathbf{r}_{k+1} &= P_{k+1}(A)\mathbf{r}_0,\\ &= a_{k+1}\left[(A+b_{k+1})P_k(A) + f_{k+1}P_{k-1}(A)\right]\mathbf{r}_0,\\ &= a_{k+1}\left[AP_k(A)\mathbf{r}_0 + b_{k+1}P_k(A)\mathbf{r}_0 + f_{k+1}P_{k-1}(A)\mathbf{r}_0\right],\\ &= a_{k+1}\left[A\mathbf{r}_k + b_{k+1}\mathbf{r}_k + f_{k+1}\mathbf{r}_{k-1}\right]. \quad (12)\end{aligned}$$

Karena vektor residu dapat juga dinyatakan dalam $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$ dan hal ini mengakibatkan

$$\begin{aligned}\mathbf{b} - A\mathbf{x}_{k+1} &= a_{k+1}\left[A\mathbf{r}_k + b_{k+1}\mathbf{r}_k + f_{k+1}\mathbf{r}_{k-1}\right],\\ &= a_{k+1}A\mathbf{r}_k + a_{k+1}\left[b_{k+1}\mathbf{r}_k + f_{k+1}\mathbf{r}_{k-1}\right],\\ &= a_{k+1}A\mathbf{r}_k + a_{k+1}b_{k+1}\mathbf{r}_k + a_k f_{k+1}\mathbf{r}_{k-1},\\ &= a_{k+1}A\mathbf{r}_k + a_{k+1}b_{k+1}\left(\mathbf{b} - A\mathbf{x}_k\right) + a_{k+1}f_{k+1}\left(\mathbf{b} - A\mathbf{x}_{k-1}\right),\\ &= a_{k+1}A\mathbf{r}_k + \mathbf{b} - a_{k+1}\left(b_{k+1}A\mathbf{x}_k + f_{k+1}A\mathbf{x}_{k-1}\right),\\ &= a_{k+1}\left[A\mathbf{r}_k - b_{k+1}A\mathbf{x}_k - f_{k+1}A\mathbf{x}_{k-1}\right] + \mathbf{b},\\ -A\mathbf{x}_{k+1} &= a_{k+1}\left[A\mathbf{r}_k - b_{k+1}A\mathbf{x}_k - f_{k+1}A\mathbf{x}_{k-1}\right]. \quad (13)\end{aligned}$$

Apabila kedua ruas persamaan (13) dikalikan dengan $-A^{-1}$ dari kiri diperoleh

$$\begin{aligned}-A^{-1}\left(-A\mathbf{x}_{k+1}\right) &= a_{k+1}\left[-A^{-1}A\mathbf{r}_k - \left(-A^{-1}b_{k+1}A\mathbf{x}_k\right) - \left(-A^{-1}f_{k+1}A\mathbf{x}_{k-1}\right)\right],\\ \mathbf{x}_{k+1} &= a_{k+1}\left[-\mathbf{r}_k + b_{k+1}I\mathbf{x}_k + f_{k+1}I\mathbf{x}_{k-1}\right],\\ \mathbf{x}_{k+1} &= a_{k+1}\left[b_{k+1}\mathbf{x}_k + f_{k+1}\mathbf{x}_{k-1} - \mathbf{r}_k\right]. \quad (14)\end{aligned}$$

Persamaan (2), (9), (10), (12), dan (14) membentuk algoritma Orthores yang disajikan sebagai berikut.

Algoritma Orthores

1. Inisialisasi $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{y}_0 = \mathbf{y}$, $f_1 = 0$, $\mathbf{x}_0 = \mathbf{x}$, $b_1 = -\dfrac{\langle \mathbf{y}_0, A\mathbf{r}_0 \rangle}{\langle \mathbf{y}_0, \mathbf{r}_0 \rangle}$,

   $a_1 = \dfrac{1}{b_1}$, $\mathbf{x}_1 = a_1(b_1\mathbf{x}_0 - \mathbf{r}_0)$ dan $\mathbf{r}_1 = a_1(A\mathbf{r}_0 + b_1\mathbf{r}_0)$.

2. Menghitung vektor residu dan solusi pendekatan

for $k = 1, 2, \ldots,$ do

$$\mathbf{y}_k = A^T \mathbf{y}_{k-1},$$

$$f_{k+1} = -\frac{\langle \mathbf{y}_k, \mathbf{r}_k \rangle}{\langle \mathbf{y}_{k-1}, \mathbf{r}_{k-1} \rangle},$$

$$b_{k+1} = -\frac{\langle \mathbf{y}_k, A\mathbf{r}_k \rangle - f_{k+1}\langle \mathbf{y}_k, \mathbf{r}_{k-1} \rangle}{\langle \mathbf{y}_k, \mathbf{r}_k \rangle},$$

$$a_{k+1} = \frac{1}{f_{k+1} + b_{k+1}},$$

$$\mathbf{r}_{k+1} = a_{k+1}(b_{k+1}\mathbf{r}_{k-1} + A\mathbf{r}_k + f_{k+1}\mathbf{r}_k),$$

$$\mathbf{x}_{k+1} = a_{k+1}(f_{k+1}\mathbf{x}_k + b_{k+1}\mathbf{x}_{k-1} - \mathbf{r}_k).$$

3.2 Hasil Numerik dan Analisis

Algoritma A_4/Orthores dapat diinterpretasikan ke dalam program komputer yang telah dimodifikasi dari (Maharani, 2015). Matriks yang digunakan dalam simulasi berbentuk matriks tridiagonal dengan pola sebagai berikut

$$A = \begin{bmatrix} B & -I & \cdots & 0 \\ -I & B & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & -I & B \end{bmatrix}, \tag{15}$$

dengan $B = \begin{bmatrix} 4 & \alpha & \cdots & \cdots & 0 \\ \beta & 4 & \alpha & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \beta & 4 & \alpha \\ 0 & \cdots & \cdots & \beta & 4 \end{bmatrix}$, $\alpha = -1 + \delta$, dan $\beta = -1 - \delta$.

Untuk mendapatkan matriks tridiagonal seperti diatas diperlukan ukuran dimensi dan nilai δ yang berbeda. Dalam penelitan ini, digunakan vektor

solusi untuk menguji kevalidan algoritma Orthores $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ n \end{bmatrix}$. Dengan demikian diperoleh vektor ruas kanan $\mathbf{b} = A\mathbf{x}$.

Pada penelitian ini juga dilakukan eksperimen untuk beberapa SPL dengan dimensi mulai dari 1000 sampai 10000. Adapun jumlah iterasi yang digunakan adalah 500, sedangkan δ yang digunakan yaitu 0; 0,3; 0,5; 0,8; 5 dan 8.

3.2.1 Hasil Numerik

Pada eksperimen ini, dilakukan sejumlah simulasi perhitungan solusi dari beberapa SPL. Algoritma Orthores diimplementasikan dalam program MATLAB R2010a, dengan menggunakan spesifikasi PC Dell inc inspiron 3847 prosesor Intel Core i3-4170 @3.70GHz, 4096 MB RAM. Seluruh hasil eksperimen disajikan pada Tabel 1 sampai 4 dan divisualisasikan dalam Gambar 1.

Tabel 1 Hasil perhitungan $\delta = 0,0$

Dimensi	b	f	A	Norm residu minimum		Norm residu terakhir		Iterasi breakdown	Waktu (detik)
				Norm residu	Iterasi	Norm residu	Iterasi		
1000	Inf	0	0	18,27	87	25,15	132	133	0,3020
2000	Inf	0	0	68,93	160	150,2	173	174	1,3118
3000	NaN	11,2197	NaN	49,3	154	79,29	352	353	6,3197
4000	NaN	1,7414	NaN	49,22	252	114,3	352	353	10,5035
5000	NaN	5,8051	NaN	40,96	169	42,56	353	354	16,0212
6000	NaN	14,2587	NaN	30,31	347	166,1	351	352	21,5595
7000	NaN	2,9947	NaN	65,67	307	82,07	352	353	28,6717
8000	NaN	-16,9049	NaN	53,64	331	86,94	352	353	36,6059
9000	NaN	272,1356	NaN	62,72	332	148,3	351	352	45,4032
10000	NaN	10,1456	NaN	293,6	156	2823	350	351	55,2944

Tabel 2 Hasil perhitungan $\delta = 0,3$

Dimensi	b	f	A	Norm residu minimum		Norm residu terakhir		Iterasi breakdown	Waktu (detik)
				Norm residu	Iterasi	Norm residu	Iterasi		
1000	0	0	0	206,2	111	206	111	112	0,2386
2000	-Inf	0	0	34,72	224	5,79E+06	232	233	1,9341
3000	NaN	-31,395	NaN	15,93	198	53,69	349	350	6,3831
4000	NaN	-Inf	NaN	158,6	108	753,1	348	349	10,3374
5000	NaN	-48,581	NaN	46,13	158	8157	347	348	15,6165
6000	NaN	-0,4035	NaN	4,167	338	4,717	350	351	21,4168
7000	NaN	1,7441	NaN	34,81	333	122,8	349	350	28,4577
8000	NaN	10,8548	NaN	349,1	347	413,1	348	349	35,9172
9000	NaN	-182,78	NaN	443	259	1101	348	349	44,9993
10000	NaN	-3,8307	NaN	61,78	284	6848	346	347	54,6536

Tabel 3 Hasil perhitungan $\delta = 0,8$

Dimensi	b	f	A	Norm residu minimum		Norm residu terakhir		Iterasi breakdown	Waktu (detik)
				Norm residu	Iterasi	Norm residu	Iterasi		
1000	NaN	NaN	NaN	0,304	162	0,9607	196	198	0,4946
2000	-Inf	0	0	1,905	179	3,394	184	185	1,4006
3000	NaN	-0,0436	NaN	64,81	292	62900	361	362	6,5319
4000	NaN	0,3609	NaN	0,5555	312	7,95	362	363	10,971
5000	NaN	5,0232	NaN	25,47	186	4250	362	363	16,3449
6000	NaN	26,8155	NaN	6,34	139	1998	362	363	22,1178
7000	NaN	-2,0518	NaN	9,327	303	27,3	362	363	29,4039
8000	NaN	0,0283	NaN	4,533	224	3059	362	363	37,3952
9000	NaN	-7,5394	NaN	0,644	342	0,9592	362	363	46,9324
10000	NaN	5,3882	NaN	0,7683	363	0,7683	362	363	57,024

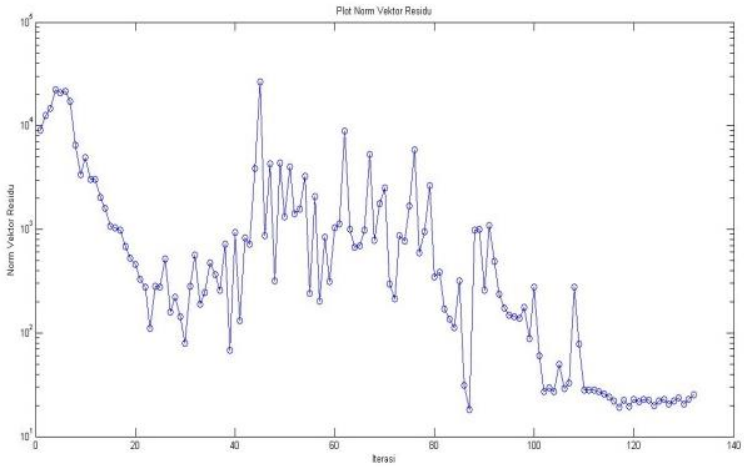
Tabel 4 Hasil perhitungan $\delta = 5$

Dimensi	b	f	A	Norm residu minimum		Norm residu terakhir		Iterasi breakdown	Waktu (detik)
				Norm residu	Iterasi	Norm residu	Iterasi		
1000	-Inf	-8	0	9,134	276	3,82E+01	297	298	0,873
2000	NaN	NaN	NaN	882,8	28	5,32E+20	288	289	2,5181
3000	NaN	-189,53	NaN	39,54	277	5,58E+02	296	297	5,0681

Dimensi	b	f	A	Norm residu minimum		Norm residu terakhir		Iterasi breakdown	Waktu (detik)
				Norm residu	Iterasi	Norm residu	Iterasi		
4000	NaN	-30,8369	NaN	2018	294	1,72E+04	295	296	8,4534
5000	NaN	-30,0417	NaN	2066	28	1,65E+08	291	292	12,7894
6000	NaN	-0,5238	NaN	2279	289	4,90E+03	295	296	17,4619
7000	NaN	6,5106	NaN	3862	27	6,88E+04	294	295	23,1904
8000	NaN	-133,872	NaN	4457	27	1,85E+07	292	293	29,3408
9000	NaN	-28,7533	NaN	5884	26	1,52E+08	291	292	36,6583
10000	NaN	-0,6341	NaN	5123	27	5,08E+09	290	291	44,8956

Keterangan :

True breakdown
Ghost breakdown



Gambar 1 Plot norm vektor residu untuk dimensi 1000 dan $\delta = 0,0$.

3.2.2 Analisis

Pada Tabel 1 sampai 4 terlihat bahwa koefisien *b*, *f* dan *a* merupakan koefisien pada iterasi breakdown terjadi, sedangkan waktu komputasi merupakan waktu yang diperlukan pada saat program berjalan. Waktu komputasi pada setiap simulasi berbeda, salah satu yang mempengaruhinya adalah besarnya dimensi. Semakin besar dimensi, maka semakin lama waktu yang dibutuhkan. Selain itu, terlihat juga norm residu terakhir tidak selalu menjadi norm residu

terbaik atau minimal. Berdasarkan Gambar 1 menunjukkan bahwa norm residu bersifat fluktuatif. Hal ini dipengaruhi oleh akumulasi error pada masing-masing iterasi.

3.3 Fenomena *Breakdown* pada Algoritma Orthores

Breakdown pada algoritma Lanczos, dalam hal ini algoritma Orthores, tidak mudah untuk dihindari. Oleh karena itu, pada sub bab ini dipaparkan mengenai breakdown yang terjadi dalam eksperimen. Kemudian diklasifikasikan apakah termasuk *true breakdown* atau *ghost breakdown*, seperti yang telah disajikan pada Tabel 1 sampai 4. Untuk lebih jelasnya, akan diilustrasikan ke dalam dua kasus berikut.

1. Kasus *True Breakdown*

Untuk kasus ini ilustrasi dilakukan dengan menggunakan $\delta = 0,0$ dan dimensi 1000. Breakdown yang terjadi pada iterasi ke-133. Kemudian, koefisien-koefisien pada iterasi ke-133 dijabarkan menjadi

$$f_{133} = \frac{0}{-1,07E+96} = 0,$$

$$b_{133} = \frac{1,78E+99}{0} = Inf,$$

$$a_{133} = \frac{1}{0+Inf} = 0.$$

Terlihat bahwa koefisien *f* dan *b* terdapat pembagian dengan nol. Karena koefisien *a* bergantung pada koefisien *f* dan *b*, maka nilai koefisien *a* menjadi 0. Lebih lanjut, koefisien-koefisien tersebut mengakibatkan vektor residu dan solusi pendekatan pada iterasi 133 menjadi NaN, sehingga terjadi *breakdown* yang menyebabkan algoritma berhenti. Jadi, *breakdown* yang terjadi termasuk kedalam *true breakdown*.

2. Kasus *Ghost breakdown*

Untuk kasus ini ilustrasi dilakukan dengan menggunakan $\delta = 0,3$ dan dimensi 10000. Breakdown yang terjadi pada iterasi ke-347. Kemudian, koefisien-koefisien pada iterasi ke-347 dijabarkan menjadi

$$f_{347} = \frac{-7,90E+291}{2,06E+291} = -3,83E+00$$

$$b_{347} = \frac{NaN}{7,90E+291} = NaN$$

$$a_{347} = \frac{1}{(-3,83E+00)+(NaN)} = NaN$$

Terlihat bahwa koefisien a bergantung pada nilai koefisien f dan b, sehingga pada saat nilai koefisien b bernilai NaN. hal ini mempengaruhi koefisien a. Selain itu, vektor residu dan solusi pendekatan pada iterasi tersebut juga tidak ada. *Breakdown* yang terjadi merupakan *ghost breakdown*.

4. KESIMPULAN DAN SARAN

4.1 Kesimpulan

1. Pada program untuk algoritma Orthores, dimensi 1000 sampai 10000 dan δ = 0,0; 0,3; 0,5; 0,8; 5 dan 8 menghasilkan solusi pendekatan dengan norm vektor residu yang fluktuatif pada setiap iterasi. Solusi pendekatan yang diperoleh bernilai NaN, karena terjadi *breakdown* pada saat proses komputasi.
2. *Breakdown* terbagi menjadi 2 macam, yaitu *true breakdown* dan *ghost breakdown*. *True breakdown* terjadi ketika terdapat pembagian dengan nol pada semua koefisien polinomial ortogonal, sedangkan *ghost breakdown* terjadi ketika terdapat pembagian dengan nol pada koefisien *b* atau *f* dan menyebabkan koefisien *a* bernilai NaN.

4.2 Saran

Kejadian *breakdown* pada salah satu algoritma Lanczos-types, yaitu A_4/Orthores telah dideteksi. Pada penelitian selanjutnya dapat dikembangkan suatu strategi untuk mengatasi terjadinya *breakdown* pada algoritma A_4/Orthores. Dengan demikian, algoritma A_4/Orthores akan menjadi algoritma yang *powerful* dalam menyelesakan SPL berdimensi tinggi.

DAFTAR PUSTAKA

Baheux, C. (1995). New implementations of Lanczos method. *Journal Of Computational And Applied Mathematics, 57*, 3-15.

Barret, R., dkk. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Philadelphia: Society for Industrial and Applied Mathematics (SIAM).

Brezinski, C., dan Sadok, H. (1993). Lanczos-type algorithms for solving systems of linear equations. *Applied Numerical Mathematics 11*, 443-473.

Brezinski, C., dan Zaglia, M. R. (1994). *Breakdown*s in the Computation of Orthogonal Polynomials. *Nonlinear Numerical Methods and Rational Approximation*, 49-59.

Brezinski, C., Zaglia, M. R., dan Sadok, H. (1992). A *Breakdown*-Free Lanczos Type Algorithm. *Numerical Mathematics, 63*, 29-38.

Brezinski, C., Zaglia, M Redivo., dan Sadok, H. (2000). The Matrix and Polynomial Approaches to Lanczos-type Algortihms. *Elsevier*, 241-260.

Farooq, M. (2011). *New Lanczos-type Algorithms and their Implementation.* Tesis. University of Essex.

Lanczos, C. (1950). An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *Journal of Research of the National Bureau of Standards, 45*, 255-282.

Lanczos, C. (1952). Solution of Systems of Linear Equations by Minimized Iterations. *Journal of Research of the National Bureau of Standards, 49*, 33-53.

Maharani. (2015). *Enhanced Lanczos Algorithms for Solving Systems of Linear Equations with Embedding Interpolation and Extrapolation*. Tesis, University of Essex.