TEKNIK REKONSTRUKSI ALJABAR UNTUK MENYELESAIKAN SISTEM PERSAMAAN LINEAR DENGAN SCILAB

RUHIYAT, M. ILYAS, A.D. GARNADI, S.NURDIATI

Departemen Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Pertanian Bogor
Jl Meranti, Kampus IPB Darmaga, Bogor 16680, Indonesia

Abstrak : Dalam artikel ini diuraikan program art.sci, sebuah program Scilab untuk menyelesaikan sistem persamaan linear berukuran sembarang. Program ini merupakan implementasi dari metode teknik rekonstruksi aljabar (TRA). Uji coba numerik TRA menggunakan sistem persamaan linear yang berasal dari diskretisasi persamaan integral jenis pertama. Salah satu obyektif tulisan ini, adalah menyediakan alat untuk menyelesaikan sistem persamaan linear sembarang di Scilab sebagai sebuah lingkungan komputasional yang bebas (free).

Katakunci: program art.sci, diskretisasi, dan Scilab

1. PENDAHULUAN

Teknik Rekonstruksi Aljabar (TRA) atau ***Algebraic Reconstruction Technique*** (ART) merupakan salah satu metode yang pertama kali digunakan diimplementasikan dalam komputer. Metode ini berupa cara iteratif yang diperkenalkan oleh S. Kaczmarz pada tahun 1937 dalam hasil karyanya yang berjudul "*Angenäherte auflösung von systemen linearer gleichungen*" (Censor, 1983).

Tujuan penulisan ini adalah menyediakan metode penyelesaian system persamaan linear berukuran bebas dalam Scilab, sebuah perangkat lunak ilmiah (Scientific software) yang bersifat bebas. Tulisan ini secara tak langsung

mendukung upaya IGOS (Indonesia Goes Open Source). Tulisan ini dibagi atas 2 bagian, pertama adalah deskripsi dari metode TRA, kemudian dilanjutkan dengan uji numerik dari metode TRA untuk menyelesaikan system persamaan linear yang diambil dari diskretisasi persamaan integral jenis pertama. Pada lampiran, diberikan pustaka numerik TRA beserta fungsi ujinya.

2 ALGORITME TEKNIK REKONSTRUKSI ALJABAR

TRA merupakan salah satu cara untuk menyelesaikan sistem persamaan linear (SPL). Misalkan SPL yang terdiri atas M persamaan dengan N faktor yang tidak diketahui adalah:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N &= b_2, \\ &\vdots \\ a_{M1}x_1 + a_{M2}x_2 + \cdots + a_{MN}x_N &= b_M. \end{aligned}$$

Persamaan linear simultan tersebut dapat dituliskan sebagai

$$\mathbf{a}_i^T \mathbf{x} = b_i, \qquad i = 1, 2, \ldots, M$$

dengan vektor kolom $\mathbf{a}_i$ dan $\mathbf{x}$ secara berturutan adalah

$$\mathbf{a}_i = \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{iN} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \qquad i = 1, 2, \ldots, M.$$

Perhatikan bahwa M persamaan ini adalah **hiperbidang** (***hyperplane***) pada ruang Euclidean berdimensi N, $\mathbb{R}^N$. Selanjutnya, setiap M persamaan ini disebut garis dan bidang secara berturut-turut untuk kasus di $\mathbb{R}^2$ dan $\mathbb{R}^3$.

SPL tidak akan mempunyai sebuah penyelesaian eksak jika M hiperbidangnya tidak memiliki perpotongan yang sama. Oleh karena itu, mencari suatu titik di dalam $\mathbb{R}^N$ yang relatif "dekat" dengan seluruh hiperbidang tersebut menjadi pilihan lain. Titik semacam ini akan menjadi sebuah penyelesaian hampiran atas SPL.

Proses iterasi pada algoritme berikut (lihat [1]) akan menghasilkan siklus rangkaian proyeksi ortogonal yang berurutan pada M hiperbidang yang dimulai dari sebarang titik awal di $\mathbb{R}^N$. Sebelumnya, diperkenalkan terlebih dahulu notasi untuk iterasi yang berurutan ini. Misalkan $\mathbf{x}_k^{(p)}$ adalah titik yang terletak pada hiperbidang ke-k yang dihasilkan saat siklus iterasi ke-p. Algoritme untuk hal ini disebut algoritme TRA sebagai berikut ini.

Algoritme TRA

1. Pilihlah titik sebarang di $\mathbb{R}^N$ dan tandai dengan $\mathbf{x}_0$.
2. Untuk siklus iterasi pertama, tetapkan $p = 1$.
3. Untuk $k = 1, 2, \ldots, M$, hitunglah
4. $$\mathbf{x}_k^{(p)} = \mathbf{x}_{k-1}^{(p)} + \frac{\left(b_k - \mathbf{a}_k^T \mathbf{x}_{k-1}^{(p)}\right)}{\mathbf{a}_k^T \mathbf{a}_k} \mathbf{a}_k.$$
5. Tetapkan $\mathbf{x}_0^{(p+1)} = \mathbf{x}_M^{(p)}$.
6. Naikkan jumlah siklus p sebanyak satu dan kembali ke Langkah 3.

Titik $\mathbf{x}_k^{(p)}$ pada langkah 3 disebut **proyeksi ortogonal** (***orthogonal projection***) dari titik $\mathbf{x}_{k-1}^{(p)}$ pada hiperbidang $\mathbf{a}_k^T \mathbf{x} = b_k$. Algoritme ini menentukan rangkaian proyeksi ortogonal yang berurutan dari satu hiperbidang ke hiperbidang berikutnya. Proyeksi akan kembali pada hiperbidang pertama setelah proyeksi pada hiperbidang terakhir dilakukan. Rumus proyeksi ortogonal disajikan pada teorema berikut.

Teorema (Rumus Proyeksi Ortogonal): Misalkan L adalah sebuah hiperbidang di dalam $\mathbb{R}^N$ dengan persamaan $\mathbf{a}^T \mathbf{x} = b$ dan misalkan $\mathbf{x}^*$ adalah sebarang titik di dalam $\mathbb{R}^N$, maka proyeksi ortogonal dari $\mathbf{x}^*$ terhadap L, yaitu $\mathbf{x}_p$, dinyatakan dengan

$$\mathbf{x}_p = \mathbf{x}^* + \frac{(b - \mathbf{a}^T \mathbf{x}^*)}{\mathbf{a}^T \mathbf{a}} \mathbf{a}.$$

Titik-titik $\mathbf{x}_M^{(1)}$, $\mathbf{x}_M^{(2)}$, $\mathbf{x}_M^{(3)}$, ... yang terletak pada hiperbidang ke-M akan konvergen menuju sebuah titik $\mathbf{x}_M^*$ pada hiperbidang tersebut (tidak tergantung pada pilihan titik awal $\mathbf{x}_0$) jika vektor-vektor $\mathbf{a}_1$, $\mathbf{a}_2$, ..., $\mathbf{a}_M$ merentang $\mathbb{R}^N$. Salah satu dari titik-titik $\mathbf{x}_M^{(p)}$ untuk p yang cukup besar akan diambil sebagai penyelesaian hampiran dari SPL.

Program SCILAB dari algoritme TRA disajikan pada Lampiran 1 dalam bentuk suatu fungsi. Nama fungsinya adalah 'tra'. Fungsi dibuat di dalam *sci-file*. Masukannya adalah matriks $A = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_M]^T$ yang berukuran $M \times N$, vektor kolom $\mathbf{b} = [b_1 \quad b_2 \quad \cdots \quad b_M]^T$ (disebut sebagai **ruas kanan** SPL) yang panjangnya M, dan vektor kolom $\mathbf{x}_0$ yang panjangnya N sebagai penyelesaian hampiran awal. Keluarannya adalah titik $\mathbf{x}_M^{(m)}$ (titik yang terletak pada hiperbidang ke-M/terakhir) yang disajikan dalam suatu vektor kolom yang panjangnya N serta bilangan m yang menyatakan banyaknya iterasi yang diperlukan. Penyelesaian hampiran ini didapatkan dengan menetapkan ukuran penghentian sebagai berikut:

$$\left\| A\mathbf{x}_M^{(m)} - \mathbf{b} \right\|_2 \leq 10^{-3}$$

dengan $\|\cdot\|_2$ menyatakan norm vektor Euclidean.

3 IMPLEMENTASI DAN HASIL KOMPUTASI

Algoritme TRA ini akan diimplementasikan untuk menyelesaikan SPL yang dibangun langsung maupun tidak langsung (dengan memodifikasi) dari matriks A dan vektor kolom $\mathbf{b}$ yang dihasilkan dari fungsi 'phillips' yang terdapat pada Lampiran 2. Matriks A yang dihasilkan merupakan matriks takdefinit negatif dan simetris. Penyelesaian yang dihasilkan dari fungsi 'phillips' akan dijadikan sebagai penyelesaian hampiran awal.

Fungsi 'phillips' merupakan fungsi yang membangkitkan matriks A, vektor kolom $\mathbf{b}$, serta vektor kolom $\mathbf{x}$ dari suatu SPL. Ketiganya dihasilkan dengan mendiskretisasi persamaan integral Fredholm jenis pertama [(Hansen, 2007)]

$$\int_{-6}^{6} \kappa(\tau,\sigma)x(\sigma)d\sigma = b(\tau), \qquad -6 \le \tau \le 6,$$

dengan Metode Galerkin. Penyelesaian x, kernel κ, dan ruas kanan b diberikan oleh

$$x(\sigma) = \begin{cases} 1+\cos\left(\frac{\pi}{3}\sigma\right), & |\sigma| < 3 \\ 0, & |\sigma| \ge 3 \end{cases}$$

$$\kappa(\tau,\sigma) = x(\tau-\sigma)$$

$$b(\tau) = (6-|\tau|)\left(1+\frac{1}{2}\cos\left(\frac{\pi}{3}\tau\right)\right)+\frac{9}{2\pi}\sin\left(\frac{\pi}{3}|\tau|\right).$$

Hal ini ditemukan oleh D. L. Phillips dan dapat dilihat pada [(Calvetti dan Reichel, 2002),(Hansen,2007)]. Masukan dari fungsi 'phillips' ini adalah bilangan bulat positif n dengan n harus kelipatan 4.

Banyaknya SPL yang akan diselesaikan ada 14 SPL. Matriks A dan vektor kolom $\mathbf{b}$ dari SPL ke-1 sampai SPL ke-10 murni dihasilkan dari fungsi 'phillips'. SPL ke-k dari sepuluh SPL pertama tersebut terdiri atas $4k$ persamaan dan $4k$ faktor yang tidak diketahui. Sebagai contoh, berikut merupakan SPL ke-1 beserta penyelesaiannya yang dibangkitkan dari fungsi 'phillips'.

$$4.2159x_1 + 0.8921x_2 = 0.4922$$
$$0.8921x_1 + 4.2159x_2 + 0.8921x_3 = 9.9001$$
$$0.8921x_2 + 4.2159x_3 + 0.8921x_4 = 9.9001$$
$$0.8921x_3 + 4.2159x_4 = 0.4922$$
$$\mathbf{x}_{phillips} = (0 \quad 1.7321 \quad 1.7321 \quad 0)^T.$$

Peyelesaian hampiran yang dihasilkan dari fungsi 'tra' ($\mathbf{x}_{tra}$) selanjutnya akan dibandingkan dengan penyelesaian yang dihasilkan dari fungsi 'phillips' ($\mathbf{x}_{phillips}$). Kedua penyelesaian dari sepuluh SPL pertama ini terdapat pada Lampiran 3. Banyaknya iterasi, *running time*, dan tingkat kesalahan dari penyelesaian sepuluh SPL pertama terdapat pada Tabel 1. Hal ini didasarkan pada ukuran penghentian proses algoritme yang telah dibahas sebelumnya.

Tabel 1. Banyaknya iterasi, *running time*, dan tingkat kesalahan untuk penyelesaian SPL ke-1 sampai SPL ke-10

SPL ke-	Banyaknya iterasi	*Running time* (detik)	$\|A\mathbf{x}_{phillips} - b\|_2$	$\|A\mathbf{x}_{tra} - b\|_2$
1	8	0.013950	2.1059	5.7613×10^{-4}
2	116	0.248873	0.8337	9.9849×10^{-4}
3	1 127	3.351584	0.3979	9.9962×10^{-4}
4	385	1.395177	0.2298	1.0000×10^{-3}
5	139	0.630695	0.1489	9.9956×10^{-4}
6	65	0.561271	0.1041	9.9243×10^{-4}
7	47	0.370987	0.0768	9.8985×10^{-4}
8	37	0.370060	0.0590	9.8878×10^{-4}
9	30	0.259787	0.0467	9.7668×10^{-4}
10	22	0.302518	0.0379	9.9003×10^{-4}

Banyaknya iterasi dan *running time* yang diperlukan untuk mendapatkan penyelesaian hampiran pada awalnya meningkat dari SPL ke-1 ke SPL ke-3, kemudian menurun dari SPL ke-3 ke SPL ke-10. Hal ini dapat dikarenakan penyelesaian hampiran awal yang diambil dari fungsi 'phillips'. Penyelesaian hampiran SPL yang dihasilkan dari fungsi 'tra' lebih akurat daripada penyelesaian yang dihasilkan dari fungsi 'phillips'. Hal ini dapat dilihat dari tingkat kesalahannya. Akan tetapi, dibutuhkan waktu lebih untuk menghasilkan penyelesaian hampiran SPL yang dihasilkan dari fungsi 'tra' ini. Sebagai contoh, berikut merupakan penyelesaian hampiran dari SPL ke-1 yang dihasilkan dari fungsi 'tra'.

$$\mathbf{x}_{tra} = (-0.3048 \quad 1.9915 \quad 1.9915 \quad -0.3048)^T.$$

Penyelesaian hampiran ini terlihat sangat berbeda dengan penyelesaian yang dihasilkan dari fungsi 'phillips'. Walaupun begitu, dapat dilihat bahwa penyelesaian yang dihasilkan dari fungsi 'phillips' semakin akurat dengan bertambah besarnya ukuran SPL.

Matriks A dari SPL ke-11 sampai SPL ke-14 murni dihasilkan dari fungsi 'phillips', sedangkan ruas kanannya dimodifikasi dari vektor kolom $\mathbf{b}$ yang dihasilkan dari fungsi 'phillips'. Ruas kanan yang digunakan adalah $\hat{\mathbf{b}} = \mathbf{b} + \mathbf{e}$, dengan $\mathbf{e}$ (disebut juga sebagai 'gangguan') merupakan vektor kolom berukuran $M \times 1$, elemen-elemennya merupakan bilangan-bilangan acak dari sebaran normal baku, dan $\|\mathbf{e}\|_2 \cong 10^{-3} \times \|\mathbf{b}\|_2$. Tujuan dari modifikasi ini adalah ingin dilihat seberapa signifikan perubahan penyelesaian hampiran apabila diberikan 'gangguan'.

SPL ke-$(10 + k)$ terdiri atas $4k$ persamaan dan $4k$ faktor yang tidak diketahui, sehingga SPL ke-$(10 + k)$ ini merupakan modifikasi dari SPL ke-k. Sebagai contoh, SPL ke-11 merupakan modifikasi dari SPL ke-1. 'Gangguan' yang diberikan yaitu vektor kolom

$$\mathbf{e} = (0.0140 \quad -0.0012 \quad 0.0138 \quad -0.0040)^T,$$

yang dibangkitkan secara acak dengan bantuan SCILAB, sehingga SPL ke-11 adalah sebagai berikut:

$$4.2159x_1 + 0.8921x_2 = 0.5062$$

$$0.8921x_1 + 4.2159x_2 + 0.8921x_3 = 9.8989$$
$$0.8921x_2 + 4.2159x_3 + 0.8921x_4 = 9.9140$$
$$0.8921x_3 + 4.2159x_4 = 0.4882.$$

Peyelesaian hampiran yang dihasilkan dari fungsi 'tra' ($\mathbf{x}_{tra}$) untuk empat SPL terakhir ini terdapat pada Lampiran 4. Banyaknya iterasi, *running time*, dan tingkat kesalahan dari penyelesaian empat SPL terakhir tersebut terdapat pada Tabel 2. Hal ini juga didasarkan pada ukuran penghentian proses algoritme yang telah dibahas sebelumnya.

Tabel 2. Banyaknya iterasi, *running time*, dan tingkat kesalahan untuk penyelesaian SPL ke-11 sampai SPL ke-14

SPL ke-	Banyaknya iterasi	*Running time* (detik)	$\|A\mathbf{x}_{tra} - b\|_2$
11	8	0.014091	0.0197
12	481	0.780851	0.0309
13	7 481	19.620956	0.0211
14	469 743	1531.209505	0.0306

Banyaknya iterasi dan *running time* yang diperlukan untuk mendapatkan penyelesaian hampiran SPL ke-12 sampai SPL ke-14 (setelah diberikan 'gangguan') jauh lebih besar dibandingkan SPL ke-2 sampai SPL ke-4 (sebelum diberikan 'gangguan'). Hal ini dapat dikarenakan penyelesaian hampiran awal yang diambil dari fungsi 'phillips' menjadi tidak efisien lagi. Sebagai contoh, berikut merupakan penyelesaian hampiran dari SPL ke-11 yang dihasilkan dari fungsi 'tra'.

$$\mathbf{x}_{tra} = (-0.3010 \quad 1.9896 \quad 1.9954 \quad -0.3064)^T.$$

Penyelesaian hampiran ini tidak terlalu berbeda dengan penyelesaian hampiran dari SPL ke-1 yang dihasilkan dari fungsi 'tra'.

4 KESIMPULAN

Penyelesaian hampiran atas SPL yang dihasilkan dengan menggunakan TRA mempunyai tingkat kesalahan yang sangat kecil. Walaupun begitu, khusus untuk SPL-SPL tertentu, penyelesaiannya membutuhkan iterasi yang banyak dan waktu yang lama. Proses mendapatkan penyelesaian hampiran atas SPL juga dipengaruhi oleh penentuan penyelesaian hampiran awalnya.

UCAPAN TERIMA KASIH

Pekerjaan in didanai oleh Kementrian Pendidkan dan Kebudayaan, Republik Indonesia, melalui "Penelitian Strategis Unggulan", hibah DIPA-IPB, 0558/023-04.2.01/12/2012, dengan kontrak no : 44/I3.24.4/SPK- PUS/IPB/ 2012.

DAFTAR PUSTAKA

[1] **Anton, H. dan C. Rorres**. 2005. *Aljabar Linear Elementer* Edisi Kedelapan Jilid 2. Jakarta: Erlangga.

[2] **Calvetti, D. dan L. Reichel**. 2002. *Tikhonov Regularization of Large Linear Problems*. BIT: 43(2): 15-16.

[3] **Censor, Y.,** 1983, *Finite series-expansion reconstruction methods*, Proceedings of the IEEE, 71(3), 409-419.

[4] **Hansen, P.C.,** 2007, *Regularization tools version 4.0 for Matlab 7.3*, Numerical Algorithms, V 46 (2), 189-194, 1017-1398

LAMPIRAN

Lampiran 1. Program SCILAB untuk algoritme TRA (untuk menyelesaikan sistem persamaan linear)

Fungsi tra

```
function [x,m] = tra(A,b,x0)
%-------------------------------------------------------------------------
--------
% Fungsi tra menghasilkan penyelesaian hampiran dari sistem linear
Ax=b :
% a11*x1 + a12*x2 + ... + a1N*xN = b1
% a21*x1 + a22*x2 + ... + a2N*xN = b2
%                    :
% aM1*x1 + aM2*x2 + ... + aMN*xN = b3
% menggunakan Teknik Rekonstruksi Aljabar.
% Penyelesaian hampiran x yang dipilih adalah x yang terletak pada
% hiperbidang ke-M.
%-------------------------------------------------------------------------
--------
% Masukan:
%   A  = matriks berukuran MxN
%        a11 a12 ... a1N
%        a21 a22 ... a2N
%                :
%        aM1 aM2 ... aMN
%   b  = vektor kolom berukuran Mx1
%        b1
%        b2
%        :
%        bM
%   x0 = vektor kolom berukuran Nx1 (penyelesaian hampiran awal)
%
% Keluaran:
%   x  = vektor kolom berukuran Nx1
%        x1
%        x2
%        :
%        xN
%   m  = banyaknya iterasi
%-------------------------------------------------------------------------
tic
M = size(A,1);
m = 0;
p = norm(A*x0-b);
if p <= 0.001
    x = x0;
else
    while p > 0.001
        m = m+1;
        for k = 1:M
            x = x0+(b(k)-A(k,:)*x0)*A(k,:)'/(A(k,:)*A(k,:)');
            x0 = x;
        end
        p = norm(A*x-b);
    end
end
toc
end
```

Lampiran 2. Program SCILAB untuk membangkitkan matriks *A*, vektor kolom **b**, dan vektor kolom **x** dari suatu SPL

Fungsi phillips

```
function [A,b,x] = phillips(n)
% PHILLIPS Test problem: Phillips "famous" problem.
%
% [A,b,x] = phillips(n)
%
% Discretization of the 'famous' first-kind Fredholm integral
% equation deviced by D. L. Phillips.  Define the function
%    phi(x) = | 1 + cos(x*pi/3) ,  |x| <  3 .
%             | 0                ,  |x| >= 3
% Then the kernel K, the solution f, and the right-hand side
% g are given by:
%    K(s,t) = phi(s-t) ,
%    f(t)   = phi(t) ,
%    g(s)   = (6-|s|)*(1+.5*cos(s*pi/3)) + 9/(2*pi)*sin(|s|*pi/3) .
% Both integration intervals are [-6,6].
%
% The order n must be a multiple of 4.

% Reference: D. L. Phillips, "A technique for the numerical solution
% of certain integral equations of the first kind", J. ACM 9
% (1962), 84-97.

% Discretized by Galerkin method with orthonormal box functions.

% Per Christian Hansen, IMM, 09/17/92.

% Check input.
if (rem(n,4)~=0), error('The order n must be a multiple of 4'), end

% Compute the matrix A.
h = 12/n; n4 = n/4; r1 = zeros(1,n);
c = cos((-1:n4)*4*pi/n);
r1(1:n4) = h + 9/(h*pi^2)*(2*c(2:n4+1) - c(1:n4) - c(3:n4+2));
r1(n4+1) = h/2 + 9/(h*pi^2)*(cos(4*pi/n)-1);
A = toeplitz(r1);

% Compute the right-hand side b.
if (nargout>1),
  b = zeros(n,1); c = pi/3;
  for i=n/2+1:n
    t1 = -6 + i*h; t2 = t1 - h;
    b(i) =   t1*(6-abs(t1)/2) ...
           + ((3-abs(t1)/2)*sin(c*t1) - 2/c*(cos(c*t1) - 1))/c ...
           - t2*(6-abs(t2)/2) ...
           - ((3-abs(t2)/2)*sin(c*t2) - 2/c*(cos(c*t2) - 1))/c;
    b(n-i+1) = b(i);
  end
  b = b/sqrt(h);
end

% Compute the solution x.
if (nargout==3)
  x = zeros(n,1);
  x(2*n4+1:3*n4) = (h + diff(sin((0:h:(3+10*eps))'*c))/c)/sqrt(h);
  x(n4+1:2*n4) = x(3*n4:-1:2*n4+1);
end
```