Kelas B

Laporan Tugas X

Kelompok A09

Analisis Numerik

Anggota Kelompok:

1. Gilang Gumilar – 1506737792
2. Gunawan – 1506757926
3. Sarah Dewi Fadlia – 1506688853
4. Stefanus James – 1506738076

Halaman Pernyataan Orisinalitas

Laporan ini adalah hasil kerja Kelompok A09 dan tidak ada bagian dari laporan yang merupakan hasil salinan (*copy*) dari kelompok atau sumber lain. Seluruh data yang dituliskan dalam laporan ini tanpa rekayasa dan dapat dipertanggungjawabkan kebenarannya. Jika kelompok kami terbukti melakukan kecurangan yang tidak semestinya kami lakukan, kami siap menerima konsekuensi yang telah diatur sebelumnya pada aturan perkuliahan Analisis Numerik Semester Genap.

.

DAFTAR ISI

BAB 1 Pendahuluan

Sistem persamaan linear banyak digunakan sebagai model untuk menyelesaikan masalah matematis. Dalam sistem persamaan linear dikelompokan menjadi 3 kategori, yaitu sistem persamaan *under-determined*, sistem persamaan bujur sangkar, sistem persamaan *over-determined*. Dalam makalah ini akan dibahas mengenai penyelesaian sistem persamaan linear bujur sangkar dengan metode numerikal dan iteratif dalam mendapatkan solusi. Dalam makalah ini algoritma yang digunakan dalam penyelesaian sistem persamaan linear secara numerik adalah faktorisasi LU dengan subtitusi gauss, sedangkan untuk pendekatan iteratif algoritma yang digunakan adalah *Successive Over-Relaxation,* yaitu merupakan penyempurnaan dari algoritma iteratif Gauss-Seidel, dengan penambahan suatu skalar relaksasi *(w)* pada setiap iterasinya yang dapat mengoptimisasi kecepatan mencapai konvergen.

BAB 2 Isi

2.1 Implementasi

2.1.1 Topik 1a: Linear System

Implementasi mencari matrix L dan U dari *banded matrix* menggunakan *partial pivoting*

```
% Fungsi ini akan mencari matrix segitiga atas (U) dan matrix segitiga
bawah (L) dari banded matrix dengan menggunakan pivoting
% Input matrix Pita A dengan lebar pita p+q+1
% Output matrix segitiga pita L (segitiga bawah) dan U (segitiga atas),
matrix permutasi p
function [L,U,p] = bandedMatrixLUPivoting(A,b)
  [n,n] = size(A);
  L = eye(n);
  p = 1:n;
  for k = 1:n-1
    %lakukan pivoting untuk mencegah division by zero untuk satu kolom
    % v = max value, idx = index pivot
    [v, idx] =  max(abs(A(k:n, k)));
% jika matrix singular
    if v == 0
      quit;
    end
    utemp = A(k, :);
    ptemp = p(k);
    A(k, :) = A(idx, :); %tukar baris diagonal dgn yg max
    p(k) = p(idx);
    A(idx, :) = utemp;
    p(idx) = ptemp;
    for i = k+1:n
      L(i,k) = A(i,k)/A(k,k);
      for j = k:n
        A(i,j) = A(i,j) - L(i,k) * A(k,j);
```

```
      endfor
    endfor
  endfor
  U = A;
endfunction
```

Kompleksitas memori:

Penghitungan memori, pada metode LU Factorization dengan partial pivoting terdapat 2 buah nilai yang perlu disimpan per iterasi, yaitu nilai dari matrix L dan nilai dari matrix U.

Kompleksitas waktu:

Terdapat 2 nested loop dari 1 loop yang berada diterluar, maka kompleksitasnya adalah $O(n^3)$.

Kelebihan: Dapat menghindari adanya division by zero

Kekurangan: Harus menyimpan indeks pertukaran baris

2.1.2 Topik 1b: Linear Systems

```
function [x, iter, runtime, w, error] = sor(A, b, w = 1,
max_it=1000, tol = 10^-6)

% init start time

start_time = cputime;

% validate input matrix A

[na, ma] = size(A);
if na != ma
  disp("Error: Matrix A should be a square matrix")
  return
endif
```

```
% validate matrix B

[nb, mb] = size(b);
if na != nb || mb != 1
  disp("Error: Matrix b should be a column matrix")
  return
endif

% check convergence condition

if !all( sum(diag(A)) >= sum(max(A),2)) || (issymmetric(A) &&
all(eig(A) > eps))
  disp("matrix input should be Strictly Diagonally Dominance or
Symmetric Definit Positive")
  disp("if not, the process is not convergent")
endif

% initial value solution for x

x = zeros(na, 1);

% updating value of x every iteration
w = 2 / (1 + sqrt(1 - cos(pi / (na + 1))^2));
i = 1;

for i=1:1000
  for j = 1:na
    sum = A(j, 1:j-1) * x(1:j-1, size(x, 2)) + A(j, j+1:na) *
x(j+1:na, i);
    x(j, i + 1) = (w / A(j,j)) * (b(j) - sum) + (1 - w) * x(j, i);
  endfor
  error = norm(b - A*x(:,i)) / norm(b); % calculate relative error
```

```
  if norm(x(:, i + 1) - x(:, i)) < tol && error < 0.1
    break;
  endif
endfor

% presenting output

runtime = cputime - start_time;
x = x( : ,i);
iter = i;
```

Algoritma *Successive Over-Relaxation* menerima input matrix A berukuran nxn dan b berukuran nx1 kemudian menginisiasi nilai dari solusi x dengan suatu nilai (dalam hal ini 0). Setelah itu, dilakukan iterasi sampai mencapai konvergen atau mencapai batas atas jumlah iterasi (1000 iterasi). Pada setiap iterasinya algoritma tersebut akan melakukan update terhadap nilai solusi x dengan formula :

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n.$$

Algoritma melakukan subtitusi nilai xi pada baris ke i hingga n, untuk mendapatkan hasil subtitusi nilai xi baris i, diperlukan penjumlahan sebanyak n-1 kolom baris tersebut, sehingga kompleksitas secara kesuluruhan adalah **O(n^2)**. Selain itu kompleksitas penyimpanan sebesar **O(m)** yaitu sebanyak jumlah m iterasi yang dilakukan, pada setiap iterasi nilai baru xi akan disimpan pada space baru. Dalam eksperimen ini iterasi yang dilakukan dibatasi hingga 1000 iterasi. Matriks yang digunakan harus memiliki sifat *diagonal dominance* dan *symmetric definit positive* untuk dapat mencapai konvergen.

2.1.3 Topik 2: Least Square Problem

Metode 1: Givens Rotation

```
function [Q,R,x] = QRgivensrotation(A,b)
```

```
%   Fungsi ini berfungsi untuk mengurai sebuah matriks A menjadi QR
sedemikian sehingga A = QR
%   Hasil dari penguraian matriks A kemudian dipakai untuk
menyelesaikan Least Square Problem Ax=b
%
%   Input:
%     Matriks A ∈ Rmxn
%     Matriks b ∈ Rmx1
%
%   Output:
%     Matriks Q ∈ Rmxm
%     Matriks segitiga atas R ∈ Rmxn ; m≥n
%     Matriks solusi x ∈ Rnx1

  [m,n] = size(A);
  Q = eye(m);
  R = A;
  P = eye(n);

  for j = 1:n
    for i = m:-1:(j+1)
      G = eye(m);
      [c,s] = givensrot( R(i-1,j),R(i,j) );
      G([i-1, i],[i-1, i]) = [c -s; s c];
      R = G'*R;
      Q = Q*G;
    end
  end
  x = P*(R\(Q'*b));
end

function [c,s] = givensrot(a,b)
```

```
%   Fungsi ini merupakan fungsi pelengkap untuk fungsi
QRgivensrotation
%   berfungsi untuk mendapatkan nilai cosinus dan sinus dari dua
nilai elemen matriks
%   Input:
%     a, b = elemen dari sebuah matriks
%
%   Output:
%     c = nilai cosinus
%     s = nilai sinus
  if b == 0
    c = 1;
    s = 0;
  elseif (abs(a) > abs(b))
    t = b/a;
    c = 1/sqrt(1+t^2);
    s = c*t;
  else
    ct = a/b;
    s = 1/sqrt(1+ct^2);
    c = s*ct;
  end
end
```

Kompleksitas memori:

Pada penghitungan memori, pada metode Givens Rotation terdapat 2 buah nilai tambahan yang perlu disimpan per iterasi, yaitu nilai cosinus dan sinus.

Kompleksitas waktu:

Pada fungsi QRgivensrotation terdapat nested loop dimana nested loop tersebut mengunjungi elemen dari baris dan kolom di bawah diagonal utama. Sehingga kompleksitasnya adalah $O(n^2)$.

Kelebihan: Dapat me nol kan elemen matriks satu per satu (introduce zeros one at a time)

Kekurangan: Kompleksitas yang lebih besar pada memory dan komputasi bila ukuran matriks yang besar.

Metode 2: Householder

```
function [Q, A] = householder(A)
  [m, n] = size(A);
  Q = eye(m);

  for i = 1:n
    v = A(i:m, i);;
    v(1) = v(1) + sign(v(1)) * norm(v);

    A(i:m, i:n) = A(i:m, i:n) - ((2 / (v' * v)) * (v * v') * A(i:m, i:n));

    Q(:, i:m) = Q(:, i:m) - (Q(:, i:m) * v) * ((2 / (v' * v)) * v');
  endfor
endfunction
```

```
function [x] = householderSolver(A, b)
  [Q, R] = householder(A);
  [m, n] = size(A);
  Qb = Q'*b;
```

```
  x(m) = Qb(m) / R(m, n);

  for i=m-1:-1:1
    sum = 0;
    for j=n:-1:i+1
      sum = sum + (R(i, j) * x(j));
    end
    x(i) = (Qb(i) - sum) / (R(i, n));
  end
endfunction
```

Kompleksitas memori:

Kompleksitas waktu: Terdapat satu loop dan perkalian matriks di dalamnya, sehingga kompleksitasnya adalah $O(n^3)$

Kelebihan: Lebih stabil karena menggunakan pencerminan untuk menolkan elemennya.

Kekurangan: Tidak dapat berjalan paralel karena setiap operasi menolkan mempengaruhi keseluruhan matriks.

2.2 Hasil Eksperimen

2.2.2 Successive Over-Relaxation

Pada penyelesaian persamaan linear Ax = b dengan menggunakan metode *Successive Over-Relaxation,* matrix yang digunakan dalam eksperimen adalah matrix yang memiliki sifat *diagonal dominance* dan *symmetric definit positive*. Berikut hasil eksperimen penyelesaian persamaan linear dengan menggunakan berbagai macam ukuran matriks persegi.

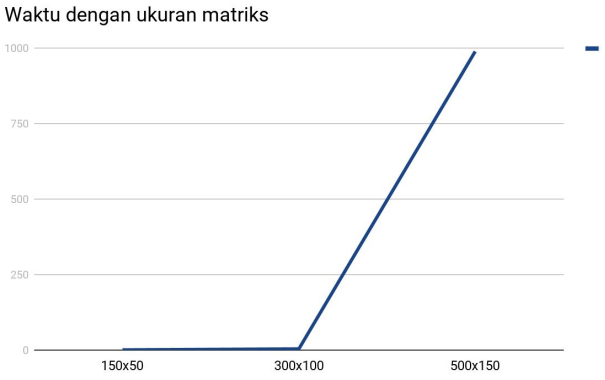
Size	Iteration	Time (ms)	omega (w)	Relative Error
5	58	0	1.333333333	3.41E-07
10	576	0	1.560387921	7.98E-07
50	1000	4.592	1.884018136	0.2251867894
100	1000	7.744	1.939676333	0.1962060265
500	1000	40.512	1.987536945	0.2888245042
1000	1000	84.592	1.99374274	0.4539800176
1500	1000	157.192	1.995822747	0.4190239385
2000	1000	200	1.996864901	0.3645312792
2500	1000	318.184	1.997490883	0.3789703224
3000	1000	416.832	1.997908493	0.4260729204

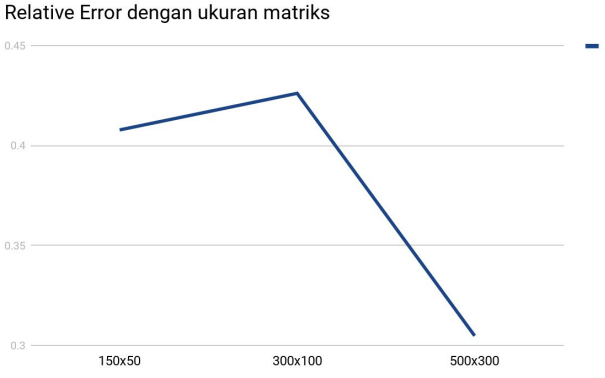
2.2.3 Topik 2: Least Square Problem

Metode: Givens Rotation

berikut adalah tabel antara ukuran matriks dan waktu yang dibutuhkan untuk dekomposisi dan mencari solusi Least Square Problem nya

Ukuran Matriks	Waktu (s)	Relative Error
150x50	0.747241	0.40779
300x100	4.0385	0.42609
500x300	988.6	0.30481





2.3 Analisis

2.3.1 Analisis Teknis

Dalam percobaan algoritma penyelesaian sistem persamaan linear dan non-linear, beberapa alat bantu yang digunakan antara lain: octave, bahasa pemrograman numerik untuk membantu pembuatan dan menjalankan algoritma, excel, membantu membuat analsis hasil eksperimen, termasuk visualisasi dengan pembuatan grafik.

2.3.2 Analisis Hasil

Berdasarkan hasil eksperimen yang dilakukan, di dapatkan hasil analisis dari setiap algoritma yang diterapkan:

Successive Over-Relaxation :

Dari hasil percobaan yang dilakukan dengan limit iterasi 1000, didapatkan akurasi untuk ukuran matrix masukan diatas 50 memiliki tingkat akurasi semakin menurun, terlihat pada nilai relatif error. Karena komputasi yang quadratic maka diperlukan jumlah iterasi yang lebih besar untuk dapat meminimumkan nilai error untuk ukuran matrik yang besar.

Least Square Problem:

Metode: Givens Rotation

Analisis yang dilakukan dari hasil percobaan mencerminkan perkiraan dari kompleksitas waktu yang ditulis sebelumnya. Dengan bertambahnya ukuran matriks, maka waktu yang dibutuhkan dengan menggunakan metode Givens Rotation ini akan bertambah sangat besar. Tingkat akurasi metode ini tidak begitu dipengaruhi oleh ukuran matriks dilihat dari relatif error hasil percobaan.

Bab 3 Penutup

3.1 Kesimpulan

Dalam penyelesaian sistem persamaan linear dan non-linear, tidak terdapat algoritma terbaik yang dapat digunakan untuk menyelesaikan semua permasalahan. Masing-masing algoritma memiliki kelebihan dan kekurangannya masing-masing, untuk itu dalam menyelesaikan permasalahan persamaan linear atau non-linear, pemilihan algoritma perlu disesuaikan dengan kebutuhan.

3.2 Log Diskusi

Diskusi 1

Hari / Tanggal	: Selasa, 13 Maret 2018
Tempat	: Fasilkom UI
Waktu	: 17.00-18.00
Agenda Diskusi	: Pembagian tugas dan pemahaman mengenai masing-masing tugas
Capaian Diskusi	: Pemahaman mengenai masing-masing tugas, mulai mengerjakan tugas

3.3 Pembagian Kerja

Contoh Tabel Pembagian Kerja

No	Nama	Tugas	Persentase (%)
1	Gunawan	Mengimplementasikan algoritma Successive Over-Relaxation	25%
2	Gilang Gumilar	Implementasi Least Square Problem Householder matrix dengan column pivoting	25%
3	Sarah Dewi Fadlia	Implementasi LU Factorization dengan Partial Pivoting	25%
4	Stefanus James	Implementasi Least Square Problem dengan givens rotation column pivoting	25%

Daftar Pustaka

"QR Decomposition Algorithm Using Givens Rotations," *matlab - QR Decomposition Algorithm Using Givens Rotations - Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/13438073/qr-decomposition-algorithm-using-givens-rotations. [Accessed: 14-Mar-2018].

"QR-factorisation using Givens-rotation. Find upper triangular matrix using Givens-rotation.," *linear algebra - QR-factorisation using Givens-rotation. Find upper triangular matrix using Givens-rotation. - Mathematics Stack Exchange*. [Online]. Available: https://math.stackexchange.com/questions/563285/qr-factorisation-using-givens-rotation-find-upper-triangular-matrix-using-given. [Accessed: 14-Mar-2018].

"Givens method," *Givens method - Algowiki*. [Online]. Available: https://algowiki-project.org/en/Givens_method. [Accessed: 15-Mar-2018].

T. Basaruddin, 2007, "Komputasi Numerik", Depok, Lembaga Penerbit Fakultas Ekonomi Universitas Indonesia.