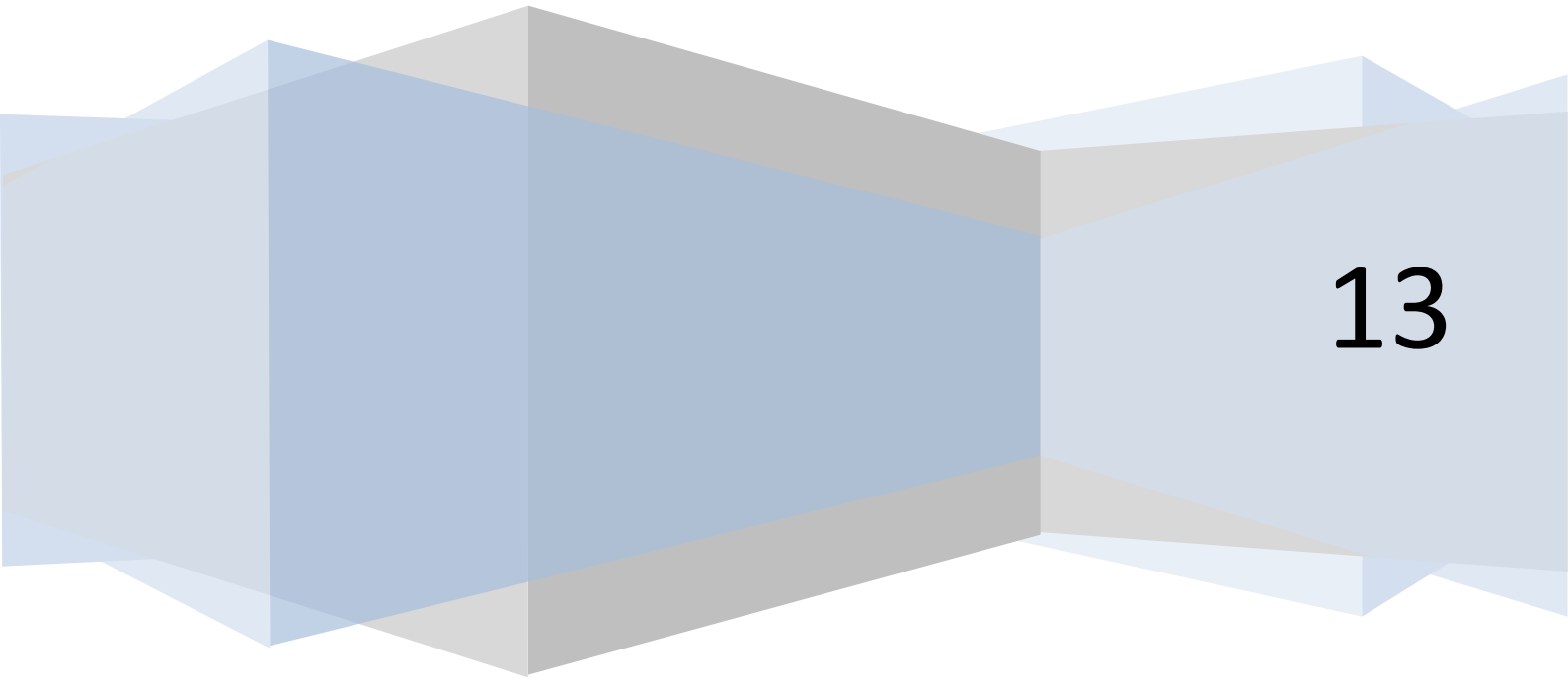
[Type the company name]

Komputer Aplikasi TE IV

Metode Numerik menggunakan Matlab

Jana Utama



13

Bab 1
Matrik dan Komputasi

Objektif :
◅ Mengenalkan matrik, vektor dan jenis-jenis matrik.
◅ Mendeklarasikan elemen-elemen matrik ke dalam memori komputer.
◅ Mengenalkan operasi penjumlahan dan perkalian matrik.
◅ Membuat *script* operasi matrik.

1.1 Mengenal matrik

Notasi suatu matrik berukuran n x m ditulis dengan huruf besar dan dicetak tebal, misalnya $\mathbf{A}_{n \times m}$. Huruf n menyatakan jumlah baris, dan huruf m jumlah kolom. Suatu matrik tersusun atas elemen-elemen yang dinyatakan dengan huruf kecil lalu diikuti oleh angka-angka indeks, misalnya a_{ij}. Indeks i menunjukkan posisi baris ke-i dan indeks j menentukan posisi kolom ke-j.

$$\mathbf{A} = (a_{ij}) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \tag{1.1}$$

Pada matrik ini, a_{11}, a_{12}, ..., a_{1m} adalah elemen-elemen yang menempati baris pertama. Sementara a_{12}, a_{22}, ..., a_{n2} adalah elemen-elemen yang menempati kolom kedua.
Contoh 1: Matrik $\mathbf{A}_{2 \times 3}$

$$\mathbf{A} = \begin{bmatrix} 3 & 8 & 5 \\ 6 & 4 & 7 \end{bmatrix}$$

dimana masing-masing elemennya adalah $a_{11} = 3$, $a_{12} = 8$, $a_{13} = 5$, $a_{21} = 6$, $a_{22} = 4$, dan $a_{23} = 7$.

Contoh 2: Matrik $\mathbf{B}_{3 \times 2}$

$$\mathbf{B} = \begin{bmatrix} 1 & 3 \\ 5 & 9 \\ 2 & 4 \end{bmatrix}$$

dimana masing-masing elemennya adalah $b_{11} = 1$, $b_{12} = 3$, $b_{21} = 5$, $b_{22} = 9$, $b_{31} = 2$, dan $b_{32} = 4$.

1.2 Vektor-baris dan vektor-kolom

Notasi vektor biasanya dinyatakan dengan huruf kecil dan dicetak tebal. Suatu matrik dinamakan vektor-baris berukuran m, bila hanya memiliki satu baris dan m kolom, yang dinyatakan sebagai berikut

$$\mathbf{a} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & \dots & a_m \end{bmatrix} \tag{1.2}$$

Sedangkan suatumatrik dinamakan vektor-kolomberukuran n, bila hanyamemiliki satu kolom dan n baris, yang dinyatakan sebagai berikut

$$\mathbf{a} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \tag{1.3}$$

1.3 Inisialisasi matrik dalam memori komputer

Sebelum dilanjutkan, saya sarankan agar anda mencari tahu sendiri bagaimana cara membuat *m-file* di Matlab dan bagaimana cara menjalankannya. Karena semua *source code* yang terdapat dalam buku ini ditulis dalam *m-file*. Walaupun sangat mudah untuk melakukan *copy-paste*, namun dalam upaya membiasakan diri menulis *source code* di *m-file*, saya anjurkan anda menulis ulang semuanya. Dalam Matlab terdapat 3 cara inisialisasi matrik. Cara pertama[1], sesuai dengan Contoh 1, adalah

```
clear all
clc

A(1,1) = 3;
A(1,2) = 8;
A(1,3) = 5;
A(2,1) = 6;
A(2,2) = 4;
A(2,3) = 7;
A
```

[1]Cara ini bisa diterapkan pada bahasa C, Fortran, Pascal, Delphi, Java, Basic, dll. Sementara cara kedua dan cara ketiga hanya akan dimengerti oleh Matlab

Sedangkan untuk matrik $\mathbf{B}_{3\times 2}$, sesuai Contoh 2 adalah

```
clear all
clc

B(1,1) = 1;
B(1,2) = 3;
B(2,1) = 5;
B(2,2) = 9;
B(3,1) = 2;
B(3,2) = 4;
B
```

Cara kedua relatif lebih mudah dan benar-benar merepresentasikan dimensi matriknya, dimana jumlah baris dan jumlah kolom terlihat dengan jelas.

```
clear all
clc

A=[ 3 8 5
   6 4 7 ];

B=[ 1 3
    5 9
    2 4 ];
A
B
```

Cara ketiga jauh lebih singkat, namun tidak menunjukkan dimensi matrik lantaran ditulis hanya dalam satu baris.

```
clear all
clc

A=[ 3 8 5 ; 6 4 7 ];
B=[ 1 3 ; 5 9 ; 2 4];
```

1.4 Macam-macam matrik

1.4.1 Matrik transpose

Operasi transpose terhadap suatumatrik akanmenukar elemen-elemen kolommenjadi elemenelemen baris. Notasi matrik tranpose adalah $\mathbf{A}^T$ atau $\mathbf{A}^t$.
Contoh 3: Operasi transpose terhadap matrik **A**

$$\mathbf{A} = \begin{bmatrix} 3 & 8 & 5 \\ 6 & 4 & 7 \end{bmatrix} \qquad \mathbf{A}^T = \begin{bmatrix} 3 & 6 \\ 8 & 4 \\ 5 & 7 \end{bmatrix}$$

Dengan Matlab, operasi transpose cukup dilakukan dengan menambahkan tanda petik tunggal di depan nama matriknya

```
clear all
clc

A=[ 3 8 5
    6 4 7 ];

AT = A';
```

1.4.2 Matrik bujursangkar

Matrik bujursangkar adalah matrik yang jumlah baris dan jumlah kolomnya sama.
Contoh 4: Matrik bujursangkar berukuran 3x3 atau sering juga disebutmatrik bujursangkar orde 3

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 8 \\ 5 & 9 & 7 \\ 2 & 4 & 6 \end{bmatrix}$$

1.4.3 Matrik simetrik

Matrik simetrik adalah matrik bujursangkar yang elemen-elemen matrik transpose-nya bernilai sama dengan matrik asli-nya.
Contoh 5: Matrik simetrik

$$\mathbf{A} = \begin{bmatrix} 2 & -3 & 7 & 1 \\ -3 & 5 & 6 & -2 \\ 7 & 6 & 9 & 8 \\ 1 & -2 & 8 & 10 \end{bmatrix} \qquad \mathbf{A}^T = \begin{bmatrix} 2 & -3 & 7 & 1 \\ -3 & 5 & 6 & -2 \\ 7 & 6 & 9 & 8 \\ 1 & -2 & 8 & 10 \end{bmatrix}$$

1.4.4 Matrik diagonal

Matrik diagonal adalah matrik bujursangkar yang seluruh elemen-nya bernilai 0 (nol), kecuali elemen-elemen diagonalnya.
Contoh 6: Matrik diagonal orde 3

$$\mathbf{A} = \begin{bmatrix} 11 & 0 & 0 \\ 0 & 29 & 0 \\ 0 & 0 & 61 \end{bmatrix}$$

1.4.5 Matrik identitas

Matrik identitas adalah matrik bujursangkar yang semua elemen-nya bernilai 0 (nol), kecuali elemen-elemen diagonal yang seluruhnya bernilai 1.
Contoh 7: Matrik identitas orde 3

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.4.6 Matrik upper-triangular

Matrik upper-tringular adalah matrik bujursangkar yang seluruh elemen dibawah elemen diagonal bernilai 0 (nol).
Contoh 8: Matrik upper-triangular

$$\mathbf{A} = \begin{bmatrix} 3 & 6 & 2 & 1 \\ 0 & 4 & 1 & 5 \\ 0 & 0 & 8 & 7 \\ 0 & 0 & 0 & 9 \end{bmatrix}$$

1.4.7 Matrik lower-triangular

Matrik lower-tringular adalah matrik bujursangkar yang seluruh elemen diatas elemen diagonal bernilai 0 (nol).
Contoh 9: Matrik lower-triangular

$$\mathbf{A} = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 32 & -2 & 0 & 0 \\ 8 & 7 & 11 & 0 \\ -5 & 10 & 6 & 9 \end{bmatrix}$$

1.4.8 Matrik tridiagonal

Matrik tridiagonal adalah matrik bujursangkar yang seluruh elemen bukan 0 (nol) berada disekitar elemen diagonal, sementara elemen lainnya bernilai 0 (nol).
Contoh 10: Matrik tridiagonal

$$\mathbf{A} = \begin{bmatrix} 3 & 6 & 0 & 0 \\ 2 & -4 & 1 & 0 \\ 0 & 5 & 8 & -7 \\ 0 & 0 & 3 & 9 \end{bmatrix}$$

1.4.9 Matrik diagonal dominan

Matrik diagonal dominan adalah matrik bujursangkar yang memenuhi

$$|a_{ii}| > \sum_{j=1, j \neq i}^{n} |a_{ij}| \qquad (1.4)$$

dimana `i`=1,2,3,..n. Coba perhatikan matrik-matrik berikut ini

$$\mathbf{A} = \begin{bmatrix} 7 & 2 & 0 \\ 3 & 5 & -1 \\ 0 & 5 & -6 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 6 & 4 & -3 \\ 4 & -2 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

Pada elemen diagonal a_{ii} matrik **A**, $|7| > |2|+|0|$, lalu $|5| > |3|+|-1|$, dan $|-6| > |5|+|0|$. Maka matrik **A** disebut matrik diagonal dominan. Sekarang perhatikan elemen diagonal matrik **B**, $|6| < |4|+|-3|$, $|-2| < |4|+|0|$, dan $|1| < |-3|+|0|$. Dengan demikian, matrik **B** bukan matrik diagonal dominan.

1.4.10 Matrik *positive-definite*

Suatu matrik dikatakan *positive-definite* bila matrik tersebut simetrik dan memenuhi

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \tag{1.5}$$

Contoh 11: Diketahui matrik simetrik berikut

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

untuk menguji apakah matrik **A** bersifat *positive-definite*, maka

$$\begin{aligned} \mathbf{x}^T\mathbf{A}\mathbf{x} &= \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2x_1 - x_2 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 \end{bmatrix} \\ &= 2x_1^2 - 2x_1x_2 + 2x_2^2 - 2x_2x_3 + 2x_3^2 \\ &= x_1^2 + (x_1^2 - 2x_1x_2 + x_2^2) + (x_2^2 - 2x_2x_3 + x_3^2) + x_3^2 \\ &= x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 \end{aligned}$$

Dari sini dapat disimpulkan bahwa matrik **A** bersifat *positive-definite*, karena memenuhi

$$x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 > 0$$

kecuali jika $x_1=x_2=x_3=0$.

1.5 Operasi matematika

1.5.1 Penjumlahan matrik

Operasi penjumlahan pada dua buah matrik hanya bisa dilakukan bila kedua matrik tersebut berukuran sama. Misalnya matrik $\mathbf{C}_{2\times3}$

$$\mathbf{C} = \begin{bmatrix} 9 & 5 & 3 \\ 7 & 2 & 1 \end{bmatrix}$$

dijumlahkan dengan matrik $\mathbf{A}_{2\times3}$, lalu hasilnya (misalnya) dinamakan matrik $\mathbf{D}_{2\times3}$

D = A + C

$$\begin{aligned}\mathbf{D} &= \begin{bmatrix}3 & 8 & 5\\ 6 & 4 & 7\end{bmatrix} + \begin{bmatrix}9 & 5 & 3\\ 7 & 2 & 1\end{bmatrix}\\ &= \begin{bmatrix}3+9 & 8+5 & 5+3\\ 6+7 & 4+2 & 7+1\end{bmatrix}\\ &= \begin{bmatrix}12 & 13 & 8\\ 13 & 6 & 8\end{bmatrix}\end{aligned}$$

Tanpamempedulikan nilai elemen-elemenmasing-masingmatrik, operasi penjumlahan antara matrik $\mathbf{A}_{2\times3}$ dan $\mathbf{C}_{2\times3}$, bisa juga dinyatakan dalam indeks masing-masing dari kedua matrik tersebut, yaitu

$$\begin{bmatrix}d_{11} & d_{12} & d_{13}\\ d_{21} & d_{22} & d_{23}\end{bmatrix} = \begin{bmatrix}a_{11}+c_{11} & a_{12}+c_{12} & a_{13}+c_{13}\\ a_{21}+c_{21} & a_{22}+c_{22} & a_{23}+c_{23}\end{bmatrix}$$

Dijabarkan satu persatu sebagai berikut

$$\begin{aligned}d_{11} &= a_{11} + c_{11}\\ d_{12} &= a_{12} + c_{12}\\ d_{13} &= a_{13} + c_{13}\\ d_{21} &= a_{21} + c_{21}\\ d_{22} &= a_{22} + c_{22}\\ d_{23} &= a_{23} + c_{23}\end{aligned} \tag{1.6}$$

Dari sini dapat diturunkan sebuah rumus umum penjumlahan dua buah matrik

$$d_{ij} = a_{ij} + c_{ij} \tag{1.7}$$

dimana `i`=1,2 dan `j`=1,2,3. **Perhatikan baik-baik! Batas `i` hanya sampai angka 2 sementara batas `j` sampai angka 3. Kemampuan anda dalam menentukan batas indeks sangat penting dalam dunia *programming*.**

1.5.2 Komputasi penjumlahan matrik

Berdasarkan contoh operasi penjumlahan di atas, indeks `j` pada persamaan (1.7) **lebih cepat** berubah dibanding indeks `i` sebagaimana ditulis pada 3 baris pertama dari Persamaan (1.6),

$$\begin{aligned}d_{11} &= a_{11} + c_{11}\\ d_{12} &= a_{12} + c_{12}\\ d_{13} &= a_{13} + c_{13}\end{aligned}$$

Jelas terlihat, ketika indeks `i` masih bernilai 1, indeks `j` sudah berubah dari nilai 1 sampai 3. Hal inimembawa konsekuensi pada *script* pemrograman, dimana *looping* untuk indeks `j` harus diletakkan di dalam *looping* indeks `i`. **Aturan mainnya adalah yang *looping*-nya paling cepat harus diletakkan paling dalam; sebaliknya, *looping* terluar adalah *looping* yang indeksnya paling jarang berubah.**

Bila anda masih belum paham terhadap kalimat yang dicetak tebal, saya akan berikan contoh *source code* dasar yang nantinya akan kita optimasi selangkah demi selangkah. OK, kita mulai dari *source code* paling mentah berikut ini.

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
D(1,1)=A(1,1)+C(1,1);
D(1,2)=A(1,2)+C(1,2);
D(1,3)=A(1,3)+C(1,3);
D(2,1)=A(2,1)+C(2,1);
D(2,2)=A(2,2)+C(2,2);
D(2,3)=A(2,3)+C(2,3);

% ---menampilkan matrik A, C dan D----
A
C
D
```

Tanda % berfungsi untuk memberikan komentar atau keterangan. Komentar atau keterangan tidak akan diproses oleh Matlab. Saya yakin anda paham dengan logika yang ada pada bagian % —*proses penjumlahan matrik*—- dalam source code di atas. Misalnya pada baris ke-9, elemen d_{11} adalah hasil penjumlahan antara elemen a_{11} dan c_{11}, sesuai dengan baris pertama Persamaan 1.6.
Tahap pertama penyederhanaan *source code* dilakukan dengan menerapkan perintah *for - end* untuk proses *looping*. *Source code* tersebut berubah menjadi

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
for j=1:3
 D(1,j)=A(1,j)+C(1,j);
 end

for j=1:3
D(2,j)=A(2,j)+C(2,j);
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

19 MARET 2013

Pada baris ke-9 dan ke-13, saya mengambil huruf j sebagai nama indeks dimana j bergerak dari 1 sampai 3. Coba anda pikirkan, mengapa j hanya bergerak dari 1 sampai 3? Modifikasi tahap kedua adalah sebagai berikut

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
i=1
for j=1:3
D(i,j)=A(i,j)+C(i,j);
end

 i=2
```

```
for j=1:3
D(i,j)=A(i,j)+C(i,j);
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Saya gunakan indeks `i` pada baris ke-9 dan ke-14 yang masing-masing berisi angka 1 dan 2. Dengan begitu indeks `i` bisa menggantikan angka 1 dan 2 yang semula ada di baris ke-11 dan ke-16. Nah sekarang coba anda perhatikan, statemen pada baris ke-10, ke-11 dan ke-12 sama persis dengan statemen pada baris ke-15, ke-16 dan ke-17, sehingga mereka bisa disatukan kedalam sebuah *looping* yang baru dimana `i` menjadi nama indeksnya.

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
for i=1:2
        for j=1:3
        D(i,j)=A(i,j)+C(i,j);
        end
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Coba anda pahami dari baris ke-9, mengapa indeks `i` hanya bergerak dari 1 sampai 2? Source code di atas memang sudah tidak perlu dimodifikasi lagi, namun ada sedikit saran untuk penulisan *looping* bertingkat dimana sebaiknya *looping* terdalam ditulis agak menjorok kedalam seperti berikut ini

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A
C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
for i=1:2
        for j=1:3
                D(i,j)=A(i,j)+C(i,j);
        end
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Sekarang anda lihat bahwa *looping* indeks `j` ditulis lebih masuk kedalam dibandingkan looping indeks `i`. Semoga contoh ini bisa memperjelas **aturan umum pemrograman dimana yang *looping*-nya paling cepat harus diletakkan paling dalam; sebaliknya, *looping* terluar adalah *looping* yang indeksnya paling jarang berubah.** Dalam contoh ini *looping* indeks `j` bergerak lebih cepat dibanding *looping* indeks `i`.

1.5.3 Perkalian matrik

Operasi perkalian dua buah matrik hanya bisa dilakukan bila jumlah kolom matrik pertama sama dengan jumlah baris matrik kedua. Jadi kedua matrik tersebut tidak harus berukuran

sama seperti pada penjumlahan dua matrik. Misalnya matrik $\mathbf{A}_{2\times3}$ dikalikan dengan matrik $\mathbf{B}_{3\times2}$, lalu hasilnya (misalnya) dinamakan matrik $\mathbf{E}_{2\times2}$

$$\mathbf{E}_{2\times2} = \mathbf{A}_{2\times3}.\mathbf{B}_{3\times2}$$

$$\begin{aligned}\mathbf{E} &= \begin{bmatrix}3 & 8 & 5\\ 6 & 4 & 7\end{bmatrix}\begin{bmatrix}1 & 3\\ 5 & 9\\ 2 & 4\end{bmatrix}\\ &= \begin{bmatrix}3.1+8.5+5.2 & 3.3+8.9+5.4\\ 6.1+4.5+7.2 & 6.3+4.9+7.4\end{bmatrix}\\ &= \begin{bmatrix}53 & 101\\ 40 & 82\end{bmatrix}\end{aligned}$$

Tanpa mempedulikan nilai elemen-elemen masing-masing matrik, operasi perkalian antara matrik $\mathbf{A}_{2\times3}$ dan $\mathbf{B}_{3\times2}$, bisa juga dinyatakan dalam indeks masing-masing dari kedua matrik tersebut, yaitu

$$\begin{bmatrix}e_{11} & e_{12}\\ e_{21} & e_{22}\end{bmatrix} = \begin{bmatrix}a_{11}.b_{11}+a_{12}.b_{21}+a_{13}.b_{31} & a_{11}.b_{12}+a_{12}.b_{22}+a_{13}.b_{32}\\ a_{21}.b_{11}+a_{22}.b_{21}+a_{23}.b_{31} & a_{21}.b_{12}+a_{22}.b_{22}+a_{23}.b_{32}\end{bmatrix}$$

Bila dijabarkan, maka elemen-elemen matrik $\mathbf{E}_{2\times2}$ adalah

$$e_{11} = a_{11}.b_{11}+a_{12}.b_{21}+a_{13}.b_{31} \quad (1.8)$$

$$e_{12} = a_{11}.b_{12}+a_{12}.b_{22}+a_{13}.b_{32} \quad (1.9)$$

$$e_{21} = a_{21}.b_{11}+a_{22}.b_{21}+a_{23}.b_{31} \quad (1.10)$$

$$e_{22} = a_{21}.b_{12}+a_{22}.b_{22}+a_{23}.b_{32} \quad (1.11)$$

Sejenak, mari kita amati perubahan pasangan angka-angka indeks yang mengiringi elemen e, elemen a dan elemen b mulai dari persamaan (1.8) sampai persamaan (1.11). Perhatikan perubahan angka-indeks-pertama pada elemen e seperti berikut ini

$$\begin{aligned}e1.. &= . .\\ e1.. &= . .\\ e2.. &= . .\\ e2.. &= . .\end{aligned}$$

Pola perubahan yang sama akan kita dapati pada angka-indeks-pertama dari elemen a

$$\begin{aligned}e1.. &= a1... b... + a1... b... + a1... b...\\ e1.. &= a1... b... + a1... b... + a1... b...\\ e2.. &= a2... b... + a2... b... + a2... b...\\ e2.. &= a2... b... + a2... b... + a2... b...\end{aligned}$$

Dengan demikian kita bisa mencantumkan huruf i sebagai pengganti angka-angka indeks yang polanya sama

$$\begin{aligned}ei.. &= ai... b... + ai... b... + ai... b...\\ ei.. &= ai... b... + ai... b... + ai... b...\\ ei.. &= ai... b... + ai... b... + ai... b...\end{aligned}$$

$$e_{i..} = a_{i...}\, b_{...} + a_{i...}\, b_{...} + a_{i...}\, b_{...}$$

dimana i bergerak mulai dari angka 1 hingga angka 2, atau kita nyatakan i=1,2. Selanjutnya, masih dari persamaan (1.8) sampai persamaan (1.11), marilah kita perhatikan perubahan angka-indeks-kedua pada elemen e dan elemen b,

$$e_{i1} = a_{i...}\, b_{..1} + a_{i...}\, b_{..1} + a_{i...}\, b_{..1}$$
$$e_{i2} = a_{i...}\, b_{..2} + a_{i...}\, b_{..2} + a_{i...}\, b_{..2}$$
$$e_{i1} = a_{i...}\, b_{..1} + a_{i...}\, b_{..1} + a_{i...}\, b_{..1}$$
$$e_{i2} = a_{i...}\, b_{..2} + a_{i...}\, b_{..2} + a_{i...}\, b_{..2}$$

Dengan demikian kita bisa mencantumkan huruf j sebagai pengganti angka-angka indeks yang polanya sama

$$e_{ij} = a_{i...}\, b_{..j} + a_{i...}\, b_{..j} + a_{i...}\, b_{..j}$$
$$e_{ij} = a_{i...}\, b_{..j} + a_{i...}\, b_{..j} + a_{i...}\, b_{..j}$$
$$e_{ij} = a_{i...}\, b_{..j} + a_{i...}\, b_{..j} + a_{i...}\, b_{..j}$$
$$e_{ij} = a_{i...}\, b_{..j} + a_{i...}\, b_{..j} + a_{i...}\, b_{..j}$$

dimana j bergerak mulai dari angka 1 hingga angka 2, atau kita nyatakan j=1,2. Selanjutnya, masih dari persamaan (1.8) sampai persamaan (1.11), mari kita perhatikan perubahan angkaindeks- kedua elemen a dan angka-indeks-pertama elemen b, dimana kita akan dapati pola sebagai berikut

$$e_{ij} = a_{i1.}\, b_{1j} + a_{i2.}\, b_{2j} + a_{i3.}\, b_{3j}$$
$$e_{ij} = a_{i1.}\, b_{1j} + a_{i2.}\, b_{2j} + a_{i3.}\, b_{3j}$$
$$e_{ij} = a_{i1.}\, b_{1j} + a_{i2.}\, b_{2j} + a_{i3.}\, b_{3j}$$
$$e_{ij} = a_{i1.}\, b_{1j} + a_{i2.}\, b_{2j} + a_{i3.}\, b_{3j}$$

Dan kita bisa mencantumkan huruf k sebagai pengganti angka-angka indeks yang polanya sama, dimana k bergerak mulai dari angka 1 hingga angka 3, atau kita nyatakan k=1,2,3.

$$e_{ij} = a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj}$$
$$e_{ij} = a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj}$$
$$e_{ij} = a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj}$$
$$e_{ij} = a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj} + a_{ik.}\, b_{kj}$$

Kemudian secara sederhana dapat ditulis sebagai berikut

$$e_{ij} = a_{ik.} b_{kj} + a_{ik.} b_{kj} + a_{ik.} b_{kj} \tag{1.12}$$

Selanjutnya dapat ditulis pula formula berikut

$$e_{ij} = \sum_{k=1}^{3} a_{ik} b_{kj} \tag{1.13}$$

dimana i=1,2; j=1,2; dan k=1,2,3.
Berdasarkan contoh ini, maka secara umum bila ada matrik $\mathbf{A}_{n\times m}$ yang dikalikan dengan matrik $\mathbf{B}_{m\times p}$, akan didapatkan matrik $\mathbf{E}_{n\times p}$ dimana elemen-elemen matrik **E** memenuhi

$$e_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj} \tag{1.14}$$

dengan i=1,2,. . . ,n; j=1,2. . . ,p; dan k=1,2. . . ,m.

1.5.4 Komputasi perkalian matrik

Mari kita mulai lagi dari *source code* paling dasar dari operasi perkalian matrik sesuai dengan contoh di atas.

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
E(1,1)=A(1,1)*B(1,1)+A(1,2)*B(2,1)+A(1,3)*B(3,1);
E(1,2)=A(1,1)*B(1,2)+A(1,2)*B(2,2)+A(1,3)*B(3,2);
E(2,1)=A(2,1)*B(1,1)+A(2,2)*B(2,1)+A(2,3)*B(3,1);
E(2,2)=A(2,1)*B(1,2)+A(2,2)*B(2,2)+A(2,3)*B(3,2);

% ---menampilkan matrik A, B dan E----
A
B
E
```

Sejenak, mari kita amati dengan cermat statemen dari baris ke-9 sampai ke-12 sambil dikaitkan dengan bentuk umum penulisan indeks pada perkalian matrik yaitu

$$e_{ij} = a_{ik}.b_{kj} + a_{ik}.b_{kj} + a_{ik}.b_{kj} \tag{1.15}$$

Dari sana ada 4 *point* yang perlu dicatat:

- elemen e memiliki indeks i dan indeks j dimana indeks j lebih cepat berubah dibanding indeks i.
- pada baris statemen ke-8 sampai ke-11 ada tiga kali operasi perkalian dan dua kali operasi penjumlahan yang semuanya melibatkan indeks i, indeks j dan indeks k. Namun indeks k selalu berubah pada masing-masing perkalian. Jadi indeks k paling cepat berubah dibanding indeks i dan indeks j.
- elemen a memiliki indeks i dan indeks k dimana indeks k lebih cepat berubah dibanding indeks i.
- elemen b memiliki indeks k dan indeks j dimana indeks k lebih cepat berubah dibanding indeks j.

Tahapan modifikasi source code perkalian matrik tidak semudah penjumlahan matrik. Dan mengajarkan logika dibalik source code perkalian matrik jauh lebih sulit daripada sekedar memodifikasi source code tersebut. Tapi akan saya coba semampu saya lewat tulisan ini walau harus perlahan-lahan. Mudah-mudahan mudah untuk dipahami.
Saya mulai denganmemecah operasi pada statemen baris ke-8 yang bertujuan menghitung nilai E(1, 1)

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
% ---E(1,1) dihitung 3 kali
E(1,1)=A(1,1)*B(1,1);
E(1,1)=E(1,1)+A(1,2)*B(2,1);
E(1,1)=E(1,1)+A(1,3)*B(3,1);
```

```
% ---E(1,2); E(2,1); dan E(2,2) masih seperti semula
E(1,2)=A(1,1)*B(1,2)+A(1,2)*B(2,2)+A(1,3)*B(3,2);
E(2,1)=A(2,1)*B(1,1)+A(2,2)*B(2,1)+A(2,3)*B(3,1);
E(2,2)=A(2,1)*B(1,2)+A(2,2)*B(2,2)+A(2,3)*B(3,2);

% ---menampilkan matrik A, B dan E----
A
B
E
```

Agar baris ke-9 memiliki pola yang sama dengan baris ke-11 dan ke-12, upaya yang dilakukan adalah

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
% ---E(1,1) dihitung 3 kali
E(1,1)=0;
E(1,1)=E(1,1)+A(1,1)*B(1,1);
E(1,1)=E(1,1)+A(1,2)*B(2,1);
E(1,1)=E(1,1)+A(1,3)*B(3,1);

% ---E(1,2); E(2,1); dan E(2,2) masih seperti semula
E(1,2)=A(1,1)*B(1,2)+A(1,2)*B(2,2)+A(1,3)*B(3,2);
E(2,1)=A(2,1)*B(1,1)+A(2,2)*B(2,1)+A(2,3)*B(3,1);
E(2,2)=A(2,1)*B(1,2)+A(2,2)*B(2,2)+A(2,3)*B(3,2);

% ---menampilkan matrik A, B dan E----
A
B
E
```

Dari sini kita bisa munculkan indeks k

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
E(1,1)=0;
for k=1:3 % k bergerak dari 1 sampai 3
        E(1,1)=E(1,1)+A(1,k)*B(k,1);
end

% ---E(1,2); E(2,1); dan E(2,2) masih seperti semula
E(1,2)=A(1,1)*B(1,2)+A(1,2)*B(2,2)+A(1,3)*B(3,2);
E(2,1)=A(2,1)*B(1,1)+A(2,2)*B(2,1)+A(2,3)*B(3,1);
E(2,2)=A(2,1)*B(1,2)+A(2,2)*B(2,2)+A(2,3)*B(3,2);

% ---menampilkan matrik A, B dan E----
A
B
E
```

Kemudian cara yang sama dilakukan pada $E(1,\ 2)$, $E(2,\ 1)$, dan $E(2,\ 2)$. Anda mesti cermat dan hati-hati dalam menulis angka-angka indeks!!!

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
E(1,1)=0;
for k=1:3
        E(1,1)=E(1,1)+A(1,k)*B(k,1);
end

E(1,2)=0;
```

```
for k=1:3
        E(1,2)=E(1,2)+A(1,k)*B(k,2);
end

E(2,1)=0;
for k=1:3
        E(2,1)=E(2,1)+A(2,k)*B(k,1);
end

E(2,2)=0;
for k=1:3
        E(2,2)=E(2,2)+A(2,k)*B(k,2);
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

26 Maret 2013

Inisialisasi elemen-elemen matrik E dengan angka nol, bisa dilakukan diawal proses perkalian yang sekaligus memunculkan indeks `i` dan `j` untuk elemen `E`

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2 % i bergerak dari 1 sampai 2
        for j=1:2 % j bergerak dari 1 sampai 2
                E(i,j)=0;
        end
end

for k=1:3
        E(1,1)=E(1,1)+A(1,k)*B(k,1);
end

for k=1:3
        E(1,2)=E(1,2)+A(1,k)*B(k,2);
end

for k=1:3
        E(2,1)=E(2,1)+A(2,k)*B(k,1);
end

for k=1:3
        E(2,2)=E(2,2)+A(2,k)*B(k,2);
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Sekarang coba anda perhatikan statemen pada baris ke-15 dan ke-19, lalu bandingkan indeks `i` dan indeks `j` pada elemen `E`. Indeks mana yang berubah? Ya. Jawabannya adalah indeks `j`. Dengan demikian kita bisa munculkan indeks `j`

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2 % i bergerak dari 1 sampai 2
        for j=1:2 % j bergerak dari 1 sampai 2
                E(i,j)=0;
        end
end

j=1;
for k=1:3
```

```
        E(1,j)=E(1,j)+A(1,k)*B(k,j);
end

j=2;
for k=1:3
        E(1,j)=E(1,j)+A(1,k)*B(k,j);
end

for k=1:3
        E(2,1)=E(2,1)+A(2,k)*B(k,1);
end

for k=1:3
        E(2,2)=E(2,2)+A(2,k)*B(k,2);
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Lihatlah, statemen dari baris ke-15 sampai ke-17 memiliki pola yang sama dengan statemen dari baris ke-20 sampai ke-22, sehingga mereka bisa disatukan kedalam *looping* indeks `j`

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2 % i bergerak dari 1 sampai 2
        for j=1:2 % j bergerak dari 1 sampai 2
                E(i,j)=0;
        end
end

for j=1:2
        for k=1:3
                E(1,j)=E(1,j)+A(1,k)*B(k,j);
        end
end

for k=1:3
        E(2,1)=E(2,1)+A(2,k)*B(k,1);
end

for k=1:3
        E(2,2)=E(2,2)+A(2,k)*B(k,2);
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Sekarang coba sekali lagi anda perhatikan statemen pada baris ke-21 dan ke-25, lalu bandingkan indeks `i` dan indeks `j` pada elemen `E`. Indeks mana yang berubah? Ya. Jawabannyatetap indeks `j`. Dengan demikian kita bisa munculkan juga indeks `j` disana

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2 % i bergerak dari 1 sampai 2
        for j=1:2 % j bergerak dari 1 sampai 2
                E(i,j)=0;
        end
end

for j=1:2
        for k=1:3
                E(1,j)=E(1,j)+A(1,k)*B(k,j);
```

```
        end
end

j=1;
for k=1:3
        E(2,j)=E(2,j)+A(2,k)*B(k,j);
end

j=2;
for k=1:3
        E(2,j)=E(2,j)+A(2,k)*B(k,j);
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Cermatilah, statemen dari baris ke-21 sampai ke-23 memiliki pola yang sama dengan statemen dari baris ke-25 sampai ke-27, sehingga mereka pun bisa disatukan kedalam *looping* indeks `j`

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2 % i bergerak dari 1 sampai 2
        for j=1:2 % j bergerak dari 1 sampai 2
                E(i,j)=0;
        end
end

for j=1:2
        for k=1:3
                E(1,j)=E(1,j)+A(1,k)*B(k,j);
        end
end

for j=1:2
        for k=1:3
                E(2,j)=E(2,j)+A(2,k)*B(k,j);
        end
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Akhirnya kita sampai pada bagian akhir tahapan modifikasi. Perhatikan baris ke-16 dan ke-22. Indeks `i` pada elemen `E` dan `A` bergerak dari 1 ke 2, sehingga indeks `i` bisa dimunculkan

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2 % i bergerak dari 1 sampai 2
        for j=1:2 % j bergerak dari 1 sampai 2
                E(i,j)=0;
        end
end

i=1;
for j=1:2
        for k=1:3
                E(i,j)=E(i,j)+A(i,k)*B(k,j);
end
end
```

```
i=2;
for j=1:2
        for k=1:3
                E(i,j)=E(i,j)+A(i,k)*B(k,j);
        end
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Sekarang, statemen dari baris ke-15 sampai ke-19 memiliki pola yang sama dengan statemen dari baris ke-22 sampai ke-26. Mereka bisa disatukan oleh *looping* indeks `i`

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2
        for j=1:2
                E(i,j)=0;
        end
end

for i=1:2
        for j=1:2
                for k=1:3
                        E(i,j)=E(i,j)+A(i,k)*B(k,j);
                end
        end
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Inilah hasil akhir dari tahapan-tahapan modifikasi yang selanjutnya saya sebut sebagai proses **optimasi**. Upaya yang baru saja saya perlihatkan, sebenarnya penuh dengan jebakan-jebakan kesalahan, terutama jika anda kurang cermat membaca indeks dan pola. Upaya seperti itu memerlukan konsentrasi dan perhatian yang tidak sebentar. Upaya semacam itu tidak semudah meng-*copy* hasil akhir optimasi. Walaupun bisa di-*copy*, namun saya menyarankan agar andamencobamelakukan proses optimasi itu sekali lagi di komputer tanpamelihat catatan ini dan tanpa bantuan orang lain. Kalau anda gagal, cobalah berfikir lebih keras untuk mencar jalan keluarnya. Jika masih tetap gagal, silakan lihat catatan ini sebentar saja sekedar untuk mencari tahu dimana letak kesalahannya. Hanya dengan cara begitu ilmu *programming* inia kan bisa menyatu pada diri anda.

1.5.5 Perkalian matrik dan vektor-kolom

Operasi perkalian antara matrik dan vektor-kolom sebenarnya sama saja dengan perkalian antara dua matrik. Hanya saja ukuran vektor-kolom boleh dibilang spesial yaitu `m x 1`, dimana`m` merupakan jumlah baris sementara jumlah kolomnya hanya satu. Misalnya matrik **A**, pada contoh 1, dikalikan dengan vektor-kolom **x** yang berukuran 3 x 1 atau disingkat dengan mengatakan vektor-kolom **x** berukuran 3, lalu hasilnya (misalnya) dinamakan vektor-kolom **y**

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

$$\mathbf{y} = \begin{bmatrix} 3 & 8 & 5 \\ 6 & 4 & 7 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$
$$= \begin{bmatrix} 3.2 + 8.3 + 5.4 \\ 6.2 + 4.3 + 7.4 \end{bmatrix}$$
$$= \begin{bmatrix} 50 \\ 52 \end{bmatrix}$$

Sekali lagi, tanpamempedulikan nilai elemen-elemenmasing-masing, operasi perkalian antara matrik **A** dan vektor-kolom **x**, bisa juga dinyatakan dalam indeksnya masing-masing, yaitu

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11}.x_1 + a_{12}.x_2 + a_{13}.x_3 \\ a_{21}.x_1 + a_{22}.x_2 + a_{23}.x_3 \end{bmatrix}$$

Bila dijabarkan, maka elemen-elemen vektor-kolom **y** adalah

$$y1 = a11.\ x1 + a12.\ x2 + a13.\ x3$$
$$y2 = a21.\ x1 + a22.\ x2 + a23.\ x3$$

kemudian secara sederhana dapat diwakili oleh rumus berikut

$$y_i = \sum_{j=1}^{3} a_{ij} x_j$$

dimana i=1,2.
Berdasarkan contoh tersebut, secara umum bila ada matrik **A** berukuran n x m yang dikalikan dengan vektor-kolom **x** berukuran m, maka akan didapatkan vektor-kolom **y** berukuran n x 1 dimana elemen-elemen vektor-kolom **y** memenuhi

$$y_i = \sum_{j=1}^{m} a_{ij} x_j \qquad (1.16)$$

dengan i=1,2,. . . ,n.

1.5.6 Komputasi perkalian matrik dan vektor-kolom

Mari kita mulai lagi dari *source code* paling dasar dari operasi perkalian antara matrik dan vektor-kolom sesuai dengan contoh di atas

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
y(1,1)=A(1,1)*x(1,1)+A(1,2)*x(2,1)+A(1,3)*x(3,1);
y(2,1)=A(2,1)*x(1,1)+A(2,2)*x(2,1)+A(2,3)*x(3,1);

% ---menampilkan matrik A, B dan E----
A
x
y
```

Sejenak, mari kita amati dengan cermat statemen dari baris ke-8 dan ke-9 sambil dikaitkan dengan bentuk umum penulisan indeks pada perkalian antara matrik dan vektor-kolom yaitu

$$y_{i1} = a_{ij}.x_{j1} + a_{ij}.x_{j1} + a_{ij}.x_{j1} \tag{1.17}$$

Dari sana ada 3 *point* yang perlu dicatat:

- elemen `y` dan elemen `x` sama-sama memiliki indeks `i` yang berpasangan dengan angka 1.
- pada baris statemen ke-8 dan ke-9 ada tiga kali operasi perkalian dan dua kali operasi penjumlahan yang semuanya melibatkan indeks `i` dan indeks `j`. Namun indeks `j` selalu berubah pada masing-masing perkalian. Jadi indeks `j` lebih cepat berubah dibanding indeks `i`.
- elemen `a` memiliki indeks `i` dan indeks `j` dimana indeks `j` lebih cepat berubah dibanding indeks `i`.

Kita mulai dengan memecah operasi pada statemen baris ke-8 yang bertujuan menghitung nilai `y`(1, 1)

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
y(1,1)=A(1,1)*x(1,1);
y(1,1)=y(1,1)+A(1,2)*x(2,1);
y(1,1)=y(1,1)+A(1,3)*x(3,1);

y(2,1)=A(2,1)*x(1,1)+A(2,2)*x(2,1)+A(2,3)*x(3,1);

% ---menampilkan matrik A, B dan E----
A
x
y
```

Agar baris ke-8 memiliki pola yang sama dengan baris ke-9 dan ke-10, upaya yang dilakukan adalah

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
y(1,1)=0;
y(1,1)=y(1,1)+A(1,1)*x(1,1);
y(1,1)=y(1,1)+A(1,2)*x(2,1);
y(1,1)=y(1,1)+A(1,3)*x(3,1);

y(2,1)=A(2,1)*x(1,1)+A(2,2)*x(2,1)+A(2,3)*x(3,1);

% ---menampilkan matrik A, B dan E----
A
x
y
```

Dari sini kita bisa munculkan indeks `j`

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
y(1,1)=0;
for j=1:3
```

```
        y(1,1)=y(1,1)+A(1,j)*x(j,1);
end

y(2,1)=A(2,1)*x(1,1)+A(2,2)*x(2,1)+A(2,3)*x(3,1);

% ---menampilkan matrik A, B dan E----
A
x
y
```

Dengan cara yang sama, baris ke-13 dimodifikasi menjadi

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
y(1,1)=0;
for j=1:3
        y(1,1)=y(1,1)+A(1,j)*x(j,1);
end

y(2,1)=0;
for j=1:3
        y(2,1)=y(2,1)+A(2,j)*x(j,1);
end

% ---menampilkan matrik A, B dan E----
A
x
y
```

Inisialisasi vektor **y** dengan angka nol dapat dilakukan diawal proses perkalian, sekaligus memunculkan indeks `i`

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
for i=1:2
       y(i,1)=0;
end

for j=1:3
        y(1,1)=y(1,1)+A(1,j)*x(j,1);
end

for j=1:3
        y(2,1)=y(2,1)+A(2,j)*x(j,1);
end

% ---menampilkan matrik A, B dan E----
A
x
y
```

Kemudian, untukmenyamakan pola statemen baris ke-13 dan ke-17, indeks `i` kembali dimunculkan

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
for i=1:2
       y(i,1)=0;
end
```

```
i=1;
for j=1:3
        y(i,1)=y(i,1)+A(i,j)*x(j,1);
end

i=2;
for j=1:3
        y(i,1)=y(i,1)+A(i,j)*x(j,1);
end

% ---menampilkan matrik A, B dan E----
A
x
y
```

Akhir dari proses optimasi adalah sebagai berikut

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
for i=1:2
        y(i,1)=0;
end

for i=1:2
        for j=1:3
                y(i,1)=y(i,1)+A(i,j)*x(j,1);
        end
end

% ---menampilkan matrik A, x dan y----
A
x
y
```

1.6 Penutup

Demikianlah catatan singkat dan sederhanamengenai jenis-jenismatrik dasar dan operasi penjumlahan dan perkalian yang seringkali dijumpai dalam pengolahan data secara numerik. Semuanya akan dijadikan acuan atau referensi pada pembahasan topik-topik numerik yang akan datang.

1.7 Latihan

Diketahui matrik **A**, matrik **B**, dan vektor **x** sebagai berikut

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & -6 & -2 \\ 5 & 9 & 7 & 5.6 \\ 2 & 4 & 8 & -1 \\ 2.3 & 1.4 & 0.8 & -2.3 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 8 & 1 & 4 & 21 \\ 3 & 10 & 5 & 0.1 \\ 7 & -2 & 9 & -5 \\ 2.7 & -12 & -8.9 & 5.7 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 0.4178 \\ -2.9587 \\ 56.3069 \\ 8.1 \end{bmatrix}$$

1. Buatlah script untuk menyelesaikan penjumlahan matrik **A** dan matrik **B**.
2. Buatlah script untuk menyelesaikan perkalian matrik **A** dan matrik **B**.
3. Buatlah script untuk menyelesaikan perkalian matrik **A** dan vektor **x**.
4. Buatlah script untuk menyelesaikan perkalian matrik **B** dan vektor **x**.

Bab 2
Fungsi

Objektif :

⊲ Mengenalkan fungsi internal.
⊲ Membuat fungsi ekstenal.
⊲ Membuat fungsi ekternal untuk penjumlahan matrik.
⊲ Membuat fungsi ekternal untuk perkalian matrik.

2.1 Fungsi internal

Pada bab terdahulu kita sudah melakukan proses optimasi penjumlahan matrik dengan *source code* akhir seperti ini

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A
C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
for i=1:2
        for j=1:3
                D(i,j)=A(i,j)+C(i,j);
         end
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Pertanyaan yang segera muncul adalah apakah *source code* tersebut bisa digunakan untuk menyelesaikan penjumlahan matrik yang dimensinya bukan 2x3 ? Misalnya

D = A + C

$$\mathbf{D} = \begin{bmatrix} 4 & 3 & 8 & 6 \\ 5 & 1 & 2 & 3 \\ 6 & 7 & 9 & 1 \end{bmatrix} + \begin{bmatrix} 2 & 6 & 7 & 2 \\ 9 & 1 & 3 & 8 \\ 5 & 8 & 4 & 7 \end{bmatrix}$$

Tentu saja bisa, asal indeks `i` bergerak dari 1 sampai 3 dan indeks `j` bergerak dari 1 sampai 4.
Lihat *source code* berikut

```
clear all
clc

A=[4 3 8 6; 5 1 2 3; 6 7 9 1]; % inisialisasi matrik A
C=[2 6 7 2; 9 1 3 8; 5 8 4 7]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
for i=1:3
        for j=1:4
                D(i,j)=A(i,j)+C(i,j);
         end
end

% ---menampilkan matrik A, C dan D----
A
```

```
C
D
```

Walaupun bisa digunakan, namun cara modifikasi seperti itu sangat tidak fleksibel dan beresiko salah jika kurang teliti. Untuk menghindari resiko kesalahan dan agar lebih fleksibel, *source code* tersebut perlu dioptimasi sedikit lagi menjadi

```
clear all
clc

A=[4 3 8 6; 5 1 2 3; 6 7 9 1]; % inisialisasi matrik A
C=[2 6 7 2; 9 1 3 8; 5 8 4 7]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
dim=size(A);
n=dim(1);
m=dim(2);
for i=1:n
        for j=1:m
                D(i,j)=A(i,j)+C(i,j);
        end
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Perhatikan, ada tambahan 3 statemen yaitu mulai dari baris ke-8 sampai ke-10. Sementara baris ke-11 dan ke-12 hanya mengalami sedikit perubahan. Statemen di baris ke-8 bermaksud mendeklarasikan variabel `dim`untuk diisi oleh hasil perhitungan fungsi internal yang bernama `size`. Matrik **A** dijadikan parameter input fungsi `size`. Fungsi `size` berguna untuk menghitung jumlah baris dan jumlah kolom dari matrik **A**. Hasilnya adalah `dim`(1) untuk jumlah baris dan `dim`(2) untuk jumlah kolom. Pada baris ke-9, variabel `n` dideklarasikan untuk menerima informasi jumlah baris dari `dim`(1), sementara variabel`m`diisi dengan informasi jumlah kolom dari `dim`(2) pada baris ke-10. Adapun baris ke-11 dan ke-12 hanya mengubah angka indeks batas atas, masing-masing menjadi `n` dan `m`. Sekarang kalau kita balik lagi menghitung penjumlahan matrik dari contoh sebelumnya yang berukuran 2x3, maka source code akan seperti ini

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A
C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
dim=size(A);
n=dim(1);
m=dim(2);
for i=1:n
        for j=1:m
                D(i,j)=A(i,j)+C(i,j);
        end
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Ajaib bukan!? Tidak ada statemen yang berubah kecuali hanya pada baris ke-4 dan ke-5. Perubahan itu tidak bisa dihindari karenamemang di kedua baris itulah deklarasi elemen-elemen matrik **A** dan matrik **C** dilakukan.

2.2 Fungsi eksternal penjumlahan matrik

Saatnya kita memasuki topik tentang pembuatan fungsi eksternal. Dari *source code* yang terakhir tadi, mari kita ambil bagian proses penjumlahan matrik-nya saja

```
dim=size(A);
n=dim(1);
m=dim(2);
for i=1:n
        for j=1:m
                D(i,j)=A(i,j)+C(i,j);
        end
end
```

Kita akan jadikan potongan *source code* ini menjadi fungsi eksternal, dengan menambahkan statemen *function* seperti ini

```
function D=jumlah(A,C)
dim=size(A);
n=dim(1);
m=dim(2);
for i=1:n
        for j=1:m
                D(i,j)=A(i,j)+C(i,j);
        end
end
```

kemudian ia harus di-*save* dengan nama *jumlah.m*. Sampai dengan langkah ini kita telah membuat fungsi eksternal dan diberi nama fungsi *jumlah*. Sederhana sekali bukan? Untuk mengujikerja fungsi eksternal tersebut, coba jalankan *source code* berikut ini

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A
C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
D=jumlah(A,C)

% ---menampilkan matrik A, C dan D----
A
C
D
```

atau anda jalankan *source code* yang berikut ini

```
clear all
clc

A=[4 3 8 6; 5 1 2 3; 6 7 9 1]; % inisialisasi matrik A
C=[2 6 7 2; 9 1 3 8; 5 8 4 7]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
D=jumlah(A,C)

% ---menampilkan matrik A, C dan D----
A
C
D
```

atau coba iseng-iseng anda ganti matrik-nya menjadi

```
clear all
clc

V=[4 3; 5 1]; % inisialisasi matrik V
W=[2 6; 9 3]; % inisialisasi matrik W

% ---proses penjumlahan matrik----
U=jumlah(V,W)
```

```
% ---menampilkan matrik V, W dan U----
W
V
U
```

09 APRIL 2013

Periksa hasilnya, betul atau salah? Pasti betul! Kesimpulannya adalah setelah fungsi eksternal berhasil anda dapatkan, maka seketika itu pula anda tidak perlu menggubrisnya lagi. Bahkan anda tidak perlu mengingat nama matrik aslinya yang tertulis di fungsi *jumlah* yaitu matrik **A**, matrik **C** dan matrik **D**. Ditambah lagi, *source code* anda menjadi terlihat lebih singkat dan elegan. Dan kini, perhatian anda bisa lebih difokuskan pada deklarasi matrik-nya saja.

2.3 Fungsi eksternal perkalian matrik

Mari kita beralih ke perkalian matrik. Kita akan membuat fungsi eksternal untuk perkalian matrik. Berikut ini adalah *source code* perkalian matrik hasil akhir optimasi yang telah ditulis panjang lebar pada bab sebelumnya

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
for i=1:2
        for j=1:2
                E(i,j)=0;
        end
end

for i=1:2
        for j=1:2
                for k=1:3
                        E(i,j)=E(i,j)+A(i,k)*B(k,j);
                end
         end
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Source code tersebut digunakan untuk menghitung perkalian matrik berikut

$$\mathbf{E}_{2\times 2} = \mathbf{A}_{2\times 3} \cdot \mathbf{B}_{3\times 2}$$

Dan kita bisa sepakati simbol indeks m, n, dan p untuk men-generalisir dimensi matrik

$$\mathbf{E}_{m\times n} = \mathbf{A}_{m\times p} \cdot \mathbf{B}_{p\times n}$$

Dengan demikian, *source code* tersebut dapat dioptimasi menjadi

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
dim=size(A);
m=dim(1);
p=dim(2);
dim=size(B);
 n=dim(2);
for i=1:m
        for j=1:n
```

```
                E(i,j)=0;
        end
end

for i=1:m
        for j=1:n
                for k=1:p
                        E(i,j)=E(i,j)+A(i,k)*B(k,j);
                end
        end
end

% ---menampilkan matrik A, B dan E----
A
B
E
```

Selanjutnya kita ambil bagian *proses perkalian matrik* nya untuk dibuat fungsi eksternal

```
function E=kali(A,B)
dim=size(A);
m=dim(1);
p=dim(2);
dim=size(B);
n=dim(2);
for i=1:m
        for j=1:n
                E(i,j)=0;
        end
end

for i=1:m
        for j=1:n
                for k=1:p
                        E(i,j)=E(i,j)+A(i,k)*B(k,j);
                end
        end
end
```

lalu di-*save* dengan nama *kali.m*, maka terciptalah fungsi eksternal yang bernama fungsi *kali*. Kemudian coba anda uji fungsi *kali* tersebut dengan menjalankan *source code* berikut

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
B = [1 3; 5 9; 2 4]; % inisialisasi matrik B

% ---proses perkalian matrik----
E = kali(A,B)

% ---menampilkan matrik A, B dan E----
A
B
E
```

Silakan anda periksa hasil perhitungannya. Pasti betul! Anda bisa mencoba perkalian matrik lainnya dengan menggunakan source code tersebut. Bahkan anda bisa mengganti nama matriknya untuk selain **A**, **B** dan **E**.

2.4 Fungsi eksternal perkalian matrik dan vektor-kolom

Mari kita beralih ke perkalian matrik dan vektor-kolom. Kita akan membuat fungsi eksternal untuk perkalian matrik dan vektor-kolom. Berikut ini adalah *source code* perkalian matrik dan vektor-kolom hasil akhir optimasi yang telah ditulis panjang lebar pada bab sebelumnya

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
```

```
for i=1:2
       y(i,1)=0;
end

for i=1:2
       for j=1:3
              y(i,1)=y(i,1)+A(i,j)*x(j,1);
       end
end

% ---menampilkan matrik A, B dan E----
A
x
y
```

Source code tersebut digunakan untuk menghitung perkalian matrik dan vektor-kolom berikut

$$\mathbf{y}_{2\times1} = \mathbf{A}_{2\times3} \cdot \mathbf{x}_{3\times1}$$

Dan kita bisa sepakati simbol indeks m dan n untuk men-generalisir dimensi matrik

$$\mathbf{y}_{m\times1} = \mathbf{A}_{m\times n} \cdot \mathbf{x}_{n\times1}$$

Dengan demikian, *source code* tersebut dapat dioptimasi menjadi

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
dim=size(A);
m=dim(1);
n=dim(2);
for i=1:m
       y(i,1)=0;
end

for i=1:m
       for j=1:n
              y(i,1)=y(i,1)+A(i,j)*x(j,1);
       end
end

% ---menampilkan matrik A, B dan E----
A
x
y
```

Selanjutnya kita ambil bagian *proses perkalian matrik dan vektor* nya untuk dibuat fungsi eksternal

```
function y=kalivektor(A,x)
dim=size(A);
m=dim(1);
n=dim(2);
for i=1:m
       y(i,1)=0;
end

for i=1:m
       for j=1:n
              y(i,1)=y(i,1)+A(i,j)*x(j,1);
       end
end
```

lalu di-*save* dengan nama *kalivektor.m*, maka terciptalah fungsi eksternal yang bernama fungsi *kalivektor*. Kemudian coba anda uji fungsi *kalivektor* tersebut dengan menjalankan *source code* berikut

```
clear all
clc

A = [3 8 5; 6 4 7]; % inisialisasi matrik A
x = [2; 3; 4]; % inisialisasi vektor x

% ---proses perkalian matrik dan vektor----
y = kalivektor(A,x);

% ---menampilkan matrik A, vektor X dan hasil perkalian Y----
A
x
y
```

Silakan anda periksa hasil perhitungannya. Pasti betul! Anda bisa mencoba perkalian matrik dan vektor-kolom dengan angka elemen yang berbeda menggunakan *source code* tersebut. Bahkan anda bisa mengganti nama matrik dan vektor nya untuk selain **A**, **x** dan **y**.

2.5 Penutup

Ada tiga pilar yang harus dikuasai oleh seorang calon *programmer*. Pertama, ia harus tahu bagaimana cara mendeklarasikan data. Kedua, ia harus tahu bagaimana mendayagunakan *flow-control*, yang dalam bab ini dan bab sebelumnya menggunakan pasangan *for-end*. Dan ketiga, ia harus bisa membuat fungsi eksternal.

Sesungguhnya Matlab memiliki banyak fungsi internal yang bisa langsung dipakai. Anda bisa coba sendiri suatu saat nanti. Kekuatan bahasa pemrograman salah satunya terletak pada seberapa kaya dia memilik banyak fungsi. *Library* adalah kata lain untuk fungsi. Jadi, suatu bahasa pemrograman akan semakin unggul bila diamemiliki semakin banyak *library*. Menurut saya, yang terdepan saat inimasih dimenangkan oleh Python. Dengan Python, *source code* anda akan bisa berjalan diWindows, Linux dan Machintos serta beberapa platform lainnya.

Bab 3
Metode Eliminasi Gauss

✍ **Objektif :**
◅ Mengenalkan sistem persamaan linear.
◅ Mengenalkan teknik triangularisasi dan substitusi mundur.
◅ Aplikasi metode Eliminasi Gauss menggunakan matrik.
◅ Membuat algoritma metode Eliminasi Gauss.
◅ Menghitung invers matrik menggunakan metode Eliminasi Gauss.

3.1 Sistem persamaan linear

Secara umum, sistem persamaan linear dinyatakan sebagai berikut

$$P_n : \qquad a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \qquad (3.1)$$

dimana a dan b merupakan konstanta, x adalah variable, n = 1, 2, 3, ...
Berikut ini adalah sistem persamaan linear yang terdiri dari empat buah persamaan yaitu P_1, P_2, P_3, dan P_4

$$\begin{array}{lrcrcrcrcl} P_1: & x_1 & + & x_2 & & & + & 3x_4 & = & 4 \\ P_2: & 2x_1 & + & x_2 & - & x_3 & + & x_4 & = & 1 \\ P_3: & 3x_1 & - & x_2 & - & x_3 & + & 2x_4 & = & -3 \\ P_4: & -x_1 & + & 2x_2 & + & 3x_3 & - & x_4 & = & 4 \end{array}$$

Problem dari sistempersamaan linear adalah bagaimana mencari nilai pengganti bagi variabel x_1, x_2, x_3, dan x_4 sehingga semua persamaan diatas menjadi benar. Langkah awal penyelesaian problem tersebut adalah dengan melakukan penyederhanaan sistem persamaan linear.

3.2 Teknik penyederhanaan

Ada banyak jalan untuk menyederhanakan sistem persamaan linear. Namun tantangannya, kita ingin agar pekerjaan ini dilakukan oleh komputer. Oleh karena itu, kita harus menciptakan algoritma yang nantinya bisa berjalan di komputer. Untuk mencapai tujuan itu, kita akan berpatokan pada tiga buah aturan operasi matematika, yaitu

- Persamaan P_i dapat dikalikan dengan sembarang konstanta λ, lalu hasilnya ditempatkan di posisi persamaan P_i. Simbol operasi ini adalah $(\lambda P_i) \rightarrow (P_i)$. Contoh

  $$P_1 : \qquad x_1 + x_2 + 3x_4 = 4$$

  jika λ = 2, maka

  $$2P_1 : \qquad 2x_1 + 2x_2 + 6x_4 = 8$$

- Persamaan P_j dapat dikalikan dengan sembarang konstanta λ kemudian dijumlahkan dengan persamaan P_i, lalu hasilnya ditempatkan di posisi persamaan P_i. Simbol operasi ini adalah $(P_i - \lambda P_j) \rightarrow (P_i)$. Contoh

$$\begin{aligned} P_2: \quad & 2x_1 + x_2 - x_3 + x_4 = 1 \\ 2P_1: \quad & 2x_1 + 2x_2 + 6x_4 = 8 \end{aligned}$$

maka operasi (P_2- $2P_1$) → (P_2) mengakibatkan perubahan pada P_2 menjadi

$$P_2: \quad -x_2 - x_3 - 5x_4 = -7$$

Dengan cara ini, maka variabel x_1 berhasil dihilangkan dari P_2. Upaya untuk menghilangkansuatu variabel merupakan tahapan penting dalam metode Eliminasi Gauss.

- Persamaan P_i dan P_j dapat bertukar posisi. Simbol operasi ini adalah (P_i).↔ (P_j). Contoh
-

$$\begin{aligned} P_2: \quad & 2x_1 + x_2 - x_3 + x_4 = 1 \\ P_3: \quad & 3x_1 - x_2 - x_3 + 2x_4 = -3 \end{aligned}$$

maka operasi (P_2) ↔ (P_3) mengakibatkan pertukaran posisi masing-masing persamaan, menjadi

$$\begin{aligned} P_2: \quad & 3x_1 - x_2 - x_3 + 2x_4 = -3 \\ P_3: \quad & 2x_1 + x_2 - x_3 + x_4 = 1 \end{aligned}$$

3.2.1 Cara menghilangkan sebuah variabel

Sebelum dilanjut, saya ingin mengajak anda untuk fokus memahami aturan operasi yang kedua. Misalnya ada 2 persamaan linear yaitu

$$\begin{aligned} P_1: \quad & 3x_1 + 2x_2 - 5x_3 + 8x_4 = 3 \\ P_2: \quad & 4x_1 + 7x_2 - x_3 + 6x_4 = 9 \end{aligned}$$

Kemudian anda diminta untuk menghilangkan variabel x_1 dari P_2. Itu artinya, anda diminta untuk memodifikasi P_2 sedemikian rupa sehingga didapat P_2 yang baru, yang didalamnya tidak ada x_1.

Berdasarkan rumus operasi (P_i-λP_j) → (P_i) , maka operasi yang tepat adalah (P_2-4/3P_1) →(P_2). Perhatikan! Bilangan λ, yaitu 4/3, harus dikalikan dengan P_1, BUKAN dengan P_2. Sedangkan angka 4/3 adalah satu-satunya angka yang bisa menghapus variabel x_1 dari P_2 lewat operasi (P_2-4/3P_1). Selengkapnya adalah sebagai berikut

$$\begin{aligned} P_2: \quad & 4x_1 + 7x_2 - x_3 + 6x_4 = 9 \\ \frac{4}{3}P_1: \quad & \frac{4}{3}3x_1 + \frac{4}{3}2x_2 - \frac{4}{3}5x_3 + \frac{4}{3}8x_4 = \frac{4}{3}3 \end{aligned}$$

Kemudian, hasil operasi (P_2-4/3P_1) disimpan sebagai P_2 yang baru

$$P_2: \quad \left(4 - \frac{4}{3}3\right)x_1 + \left(7 - \frac{4}{3}2\right)x_2 - \left(1 - \frac{4}{3}5\right)x_3 + \left(6 - \frac{4}{3}8\right)x_4 = \left(9 - \frac{4}{3}3\right)$$

Dengan sendirinya x_1 akan lenyap dari P2. Mudah-mudahan jelas sampai disini. Demikianlah cara untuk menghilangkan x_1 dari P2.

3.2.2 Permainan indeks

Sekarang, mari kita tinjau hal yang sama, yaitu menghilangkan x_1 dari P_2, namun menggunakan 'permainan' indeks. Secara umum, P_1 dan P_2 bisa dinyatakan sebagai

$$\begin{aligned} P_1: \quad & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = a_{15} \\ P_2: \quad & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = a_{25} \end{aligned}$$

Agar x_1 hilang dari P_2, operasi yang benar adalah $(P_2 - \lambda P_1) \rightarrow (P_2)$, dimana $\lambda = \frac{a21}{a11}$. Dengan demikian, P_2 yang baru akan memenuhi

$$P_2: \left(a_{21} - \frac{a_{21}}{a_{11}}a_{11}\right)x_1 + \left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \left(a_{23} - \frac{a_{21}}{a_{11}}a_{13}\right)x_3 + \left(a_{24} - \frac{a_{21}}{a_{11}}a_{14}\right)x_4 = \left(a_{25} - \frac{a_{21}}{a_{11}}a_{15}\right)$$

Perhatikanlah variasi indeks pada persamaan diatas. Semoga intuisi anda bisa menangkap keberadaan suatu pola perubahan indeks. Jika belum, mari kita kembangkan persoalan ini. Sekarang saya ketengahkan kehadapan anda tiga buah persamaan, yaitu P_1, P_2 dan P_3

$$\begin{aligned} P_1: \quad & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = a_{15} \\ P_2: \quad & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = a_{25} \\ P_3: \quad & a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = a_{35} \end{aligned}$$

Bagaimana cara menghilangkan x_1 dari P_3 dengan memanfaatkan P_1??
Begini caranya, $(P_3 - \lambda P_1) \rightarrow (P_3)$, dengan $\lambda = \frac{a31}{a11}$

$$P_3: \left(a_{31} - \frac{a_{31}}{a_{11}}a_{11}\right)x_1 + \left(a_{32} - \frac{a_{31}}{a_{11}}a_{12}\right)x_2 + \left(a_{33} - \frac{a_{31}}{a_{11}}a_{13}\right)x_3 + \left(a_{34} - \frac{a_{31}}{a_{11}}a_{14}\right)x_4 = \left(a_{35} - \frac{a_{31}}{a_{11}}a_{15}\right)$$

Mudah-mudahan, pola perubahan indeksnya semakin jelas terlihat. Selanjutnya jika ada persamaan P_4 yang ingin dihilangkan x_1 nya dengan memanfaatkan P, bagaimana caranya? Tentu saja operasinya adalah $(P_4 - \lambda P_1) \rightarrow (P_4)$, dengan $\lambda = \frac{a41}{a11}$

$$P_4: \left(a_{41} - \frac{a_{41}}{a_{11}}a_{11}\right)x_1 + \left(a_{42} - \frac{a_{41}}{a_{11}}a_{12}\right)x_2 + \left(a_{43} - \frac{a_{41}}{a_{11}}a_{13}\right)x_3 + \left(a_{44} - \frac{a_{41}}{a_{11}}a_{14}\right)x_4 = \left(a_{45} - \frac{a_{41}}{a_{11}}a_{15}\right)$$

3.3 Triangularisasi dan Substitusi Mundur

3.3.1 Contoh pertama

Sekarang, mari kita kembali kepada sistem persamaan linear yang sudah ditulis di awal bab Ini

$$\begin{aligned} P_1: \quad & x_1 + x_2 + 3x_4 = 4 \\ P_2: \quad & 2x_1 + x_2 - x_3 + x_4 = 1 \\ P_3: \quad & 3x_1 - x_2 - x_3 + 2x_4 = -3 \\ P_4: \quad & -x_1 + 2x_2 + 3x_3 - x_4 = 4 \end{aligned}$$

Sekali lagi saya tegaskan bahwa problem dari sistem persamaan linear adalah bagaimana mendapatkan angka-angka yang bisa menggantikan variabel x_1, x_2, x_3, dan x_4 sehingga semuapersamaan di atas menjadi benar. Dengan berpegang pada ketiga teknik penyederhanaan tadi, maka sistem persamaan linear di atas dapat disederhanakan dengan langkah-langkah berikut ini:

1. Gunakan persamaan P_1 untuk menghilangkan variabel x_1 dari persamaan P_1, P_2, P_3 dan P_4 dengan cara $(P_2 - 2P_1) \rightarrow (P_2)$, $(P_3 - 3P_1) \rightarrow (P_3)$ dan $(P_4 + P_1) \rightarrow (P_4)$. Hasilnya akan seperti ini

$$\begin{aligned} P_1: \quad & x_1 + x_2 + 3x_4 = 4, \\ P_2: \quad & -x_2 - x_3 - 5x_4 = -7, \\ P_3: \quad & -4x_2 - x_3 - 7x_4 = -15, \\ P_4: \quad & 3x_2 + 3x_3 + 2x_4 = 8 \end{aligned}$$

   Silakan anda cermati bahwa x_1 kini telah hilang dari P_2, P_3 dan P_4

   .

2. Gunakan persamaan P_2 untuk menghilangkan variabel x_2 dari persamaan P_3 dan P_4 dengan cara $(P_3 - 4P_2) \rightarrow (P_3)$ dan $(P_4 + 3P_2) \rightarrow (P_4)$. Hasilnya akan seperti ini

$$\begin{aligned} P_1: \quad & x_1 + x_2 + 3x_4 = 4, \\ P_2: \quad & -x_2 - x_3 - 5x_4 = -7, \\ P_3: \quad & 3x_3 + 13x_4 = 13, \\ P_4: \quad & -13x_4 = -13 \end{aligned}$$

   Kalau x_3 masih ada di persamaan P_4, dibutuhkan satu operasi lagi untuk menghilangkannya. Namun hasil operasi pada langkah ke-2 ternyata sudah otomatis menghilangkan x_3 dari P_4. Bentuk akhir dari sistem persamaan linear di atas, dikenal sebagai bentuk **triangular**.
   Sampai dengan langkah ke-2 ini, kita berhasil mendapatkan sistem persamaan linear yang lebih sederhana. Apa yang dimaksud dengan sederhana dalam konteks ini? Suatu sistem persamaan linear dikatakan sederhana bila kita bisa mendapatkan seluruh nilai pengganti variabelnya dengan cara yang lebih mudah atau dengan usaha yang tidak memakan waktu lama dibandingkan sebelum disederhanakan.

3. Selanjutnya kita jalankan proses **backward-substitution** untuk mendapatkan angka-angka pengganti bagi x_1, x_2, $x_3$dan x_4. Melalui proses **backward-substitution**, yang pertama kali didapat adalah angka pengganti bagi variabel x_4, kemudian x_3, lalu diikuti x_2, dan akhirnya x_1. Silakan cermati yang berikut ini

$$\begin{aligned} P_4: \quad & x_4 = \frac{-13}{-13} = 1, \\ P_3: \quad & x_3 = \frac{1}{3}(13 - 13x_4) = \frac{1}{3}(13 - 13) = 0, \\ P_2: \quad & x_2 = -(-7 + 5x_4 + x_3) = -(-7 + 5 + 0) = 2, \\ P_1: \quad & x_1 = 4 - 3x_4 - x_2 = 4 - 3 - 2 = -1 \end{aligned}$$

Jadi solusinya adalah $x_1 = -1$, $x_2 = 2$, $x_3 = 0$ dan $x_4 = 1$. Coba sekarang anda cek, apakah semua solusi ini cocok dan tepat bila dimasukan ke sistempersamaan linear yang pertama, yaitu yang belum disederhanakan?
OK, mudah-mudahan ngerti ya... Kalau belum paham, coba dibaca sekali lagi. Atau, sekarang kita beralih kecontoh yang lain.

3.3.2 Contoh kedua

Diketahui sistem persamaan linear, terdiri dari empat buah persamaan yaitu P_1, P_2, P_3, dan P_4 seperti berikut ini:

$$\begin{array}{lrcrcrcrl} P_1: & x_1 & - & x_2 & + & 2x_3 & - & x_4 & = & -8 \\ P_2: & 2x_1 & - & 2x_2 & + & 3x_3 & - & 3x_4 & = & -20 \\ P_3: & x_1 & + & x_2 & + & x_3 & & & = & -2 \\ P_4: & x_1 & - & x_2 & + & 4x_3 & + & 3x_4 & = & 4 \end{array}$$

Seperti contoh pertama, solusi sistempersamaan linear di atas akan dicari dengan langkah-langkah berikut ini:

1. Gunakan persamaan P_1 untuk menghilangkan x_1 dari persamaan P_2, P_3, P_4 dengan cara $(P_2\text{-}2P_1) \rightarrow (P_2)$, $(P_3\text{-}3P_1) \rightarrow (P_3)$ dan $(P_4\text{-}4P_1) \rightarrow (P_4)$. Hasilnya akan seperti ini
2.

4. asa

Daftar Pustaka

Suparno, Supriyanto., “Komputasi untuk Sains dan Teknik –Menggunakan Matlab-“. *Edisi III Revisi terakhir tgl: 29 Juni 2010*