LINEAR ALGEBRAIC EQUATION

Menghitung nilai x1, x2, ... xn yang memenuhi beberapa persamaan sekaligus

Untuk kasus tertentu, sistem persamaan linear dapat berupa:

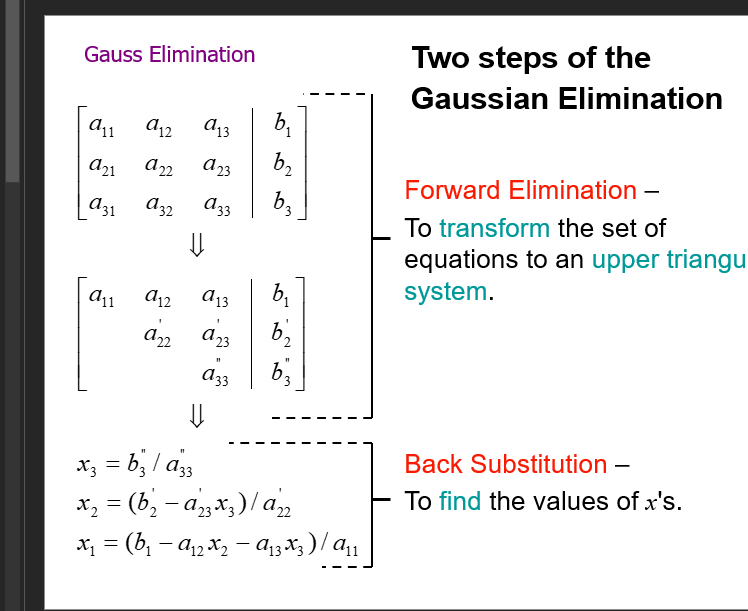
* No solution
* Infinite solution
* Ill-conditioned system

Metode2 untuk menyelesaikan sistem persamaan linear

=> Naive gauss elimination 2n^3/3 + O(n^2)

Dikatakan naïve karena disini row tidak ditukar, seingga terkadang muncul masalah error karena pembagian dengan 0 oleh komputer.

Terdiri dari 2 step: “forward elimination” dan “back subtitution”



Kekurangan eliminasi gauss:

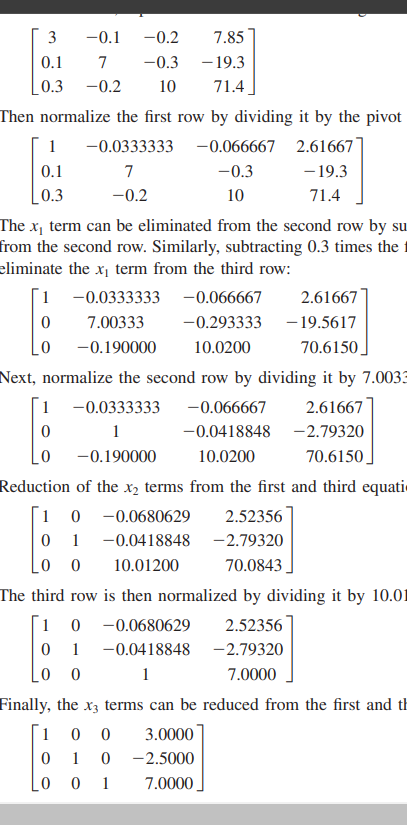
* Division by zero
* Round-off-error
* Ill-conditioned system

Cara mengoptimisasi eliminasi gauss:

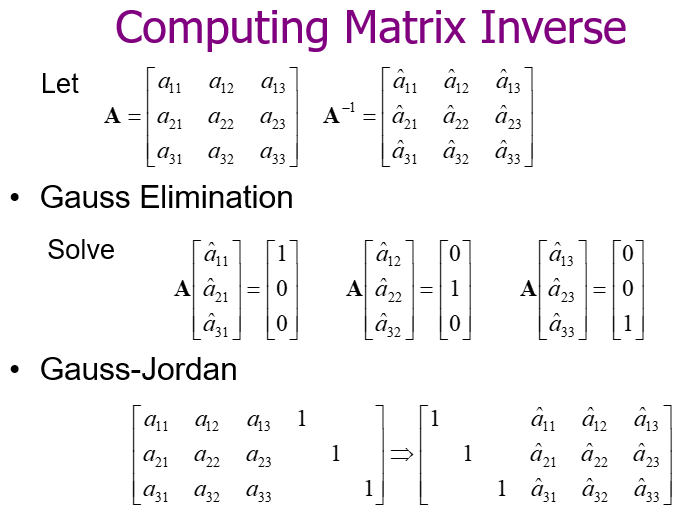
* Menambah angka penting decimal (menghindari round-off error)
* Pivoting (menghindari division by zero)
* Scaling (men-scale sehingga setiap baris memiliki koefisien terbesar 1)

=> Gauss-Jordan n^3 + O(n^2) atau 4n^3/3 (kata ppt)

Perbedaannya dari gauss elimination adalah setiap row akan dipivoting terlebih dahulu, dan persamaan yg di “nol” kan tidak hanya yg bawah namun juga yang atas

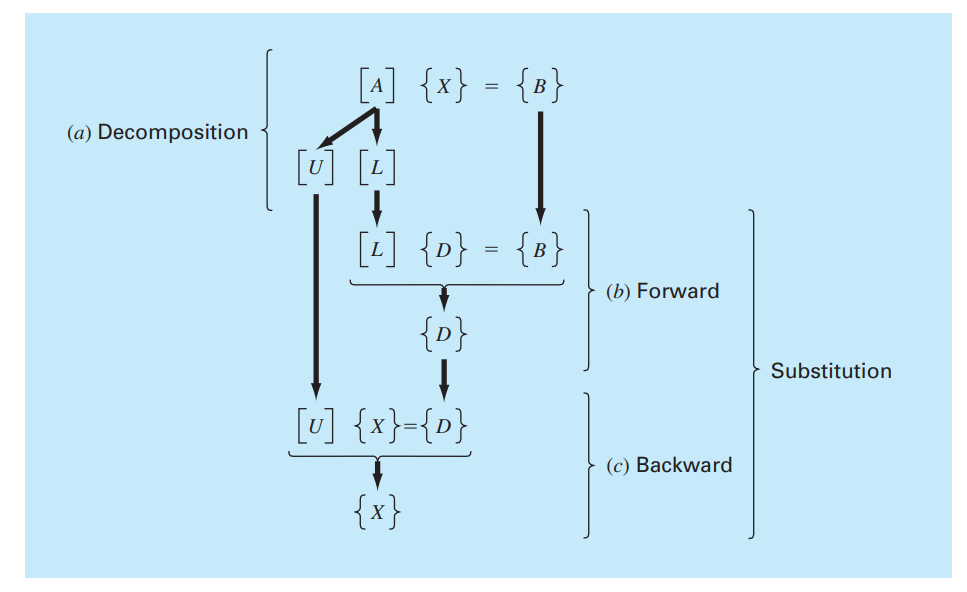
Ilustrasi: sehingga di gauss-jordan tdk diperlukan “back substitution” seperti di naïve gauss

++ how gauss m gauss-jordan compute inverse



* LU Decomposition n^3/3 + O(n^2) atau O(n^3) + K\*O(n^2)

Metode menyelesaikan sistem persamaan dengan memecah matriks A menjadi LU. Step2 nya adalah 1. LU decomposition step 2. Substitution step



* Crout Decomposition

LU decomposition (Doolittle decomposition) menciptakan 2 matriks L dan U dimana matriks L memiliki nilai 1 di diagonalnya. Crout decomposition serupa dengan LU decomposition, hanya saja matriks yang memiliki 1 di diagonalnya adalah matriks U

* Inverse Matrix 4n^3/3

Inverse matrix dengan LU decomposition [A]=[L][U] dapat dilakukan dengan menghitung semua {D} dimana [L]{D} = {B} dan {B} di set sebagai unit vector {1 0 0} (0 1 0} {0 0 1} secara bergantian. lalu menghitung {X} di persamaan [U]{X}={D} untuk setiap {D} yang terhitung, lalu hasilnya digabungkan menjadi matrix [{X1} {X2} ... {Xn}] yang merupakan inverse dari matriks A

[L]{D1}={1 0 0} [L]{D2}={0 1 0} [L]{D3}={0 0 1}

[U]{X1}={D1} [U]{X2}={D2} [U]{X3}={D3}

Maka A^-1 = [{X1} {X2} {X3}]

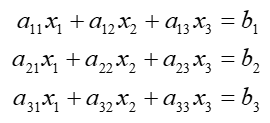
Note: mendeteksi ill-conditioned system dengan matriks:

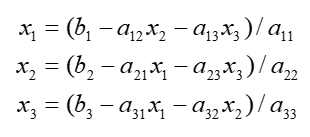
* Lakukan scale untuk setiap baris [A] sehingga nilai elemen tertingginya adalah 1, lalu lakukan inverse. Apabila elemen dari [A]^-1 memiliki nilai yang tinggi (100 ke atas), maka kemungkinan sistem [A] ill-conditioned
* Kalikan matriks A dengan hasil hitungan inversenya. Apabila hasilnya tidak mendekati matriks identitas, maka A ill-conditioned
* Lakukan inverse kembali untuk hasil inverse A, apabila hasilnya tidak cukup dekat dengan A, maka A ill-conditioned

**ITERATIVE METHOD**

Ide:

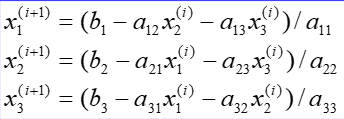
Karena:

Maka: 

Sehingga: 

yg mana merupakan dasar ide dari iterative method

* Jacobi Iteration

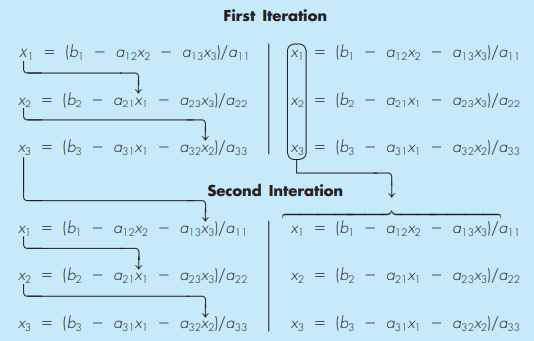
Step1: buat persamaan  berdasar parameter matrix

Step2: mulai dari initial-guess (biasanya x1(0) = x2(0) = x3(0) = 0)

Step3: hitung Kembali x1(i+1) , x2(i+1) , x3(i+1) berdasar formula yang telah didefinisikan dengan i=0,1,2,… sampai semua x konvergen ke suatu nilai.

* Gauss-Seidel Iteration

Ialah improvisasi dari jacobi iteration. Alih-alih menghitung xn(i+1) secara bebarengan seperti di jacobi, iterasi gauss-seidel menghitung xn(i+1) satu-satu dan hasilnya langsung disubstitusikan ke persamaan selanjutnya

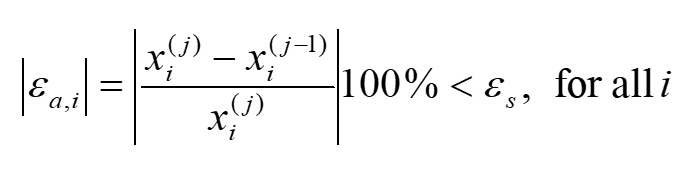


Gauss-Seidel (kiri); Jacobi (kanan)

Note:

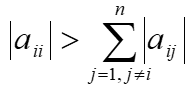
* stopping criteria

Iterasi jacobi dan gauss-seidel akan di stop apabila error relativenya cukup kecil



* konvergensi

metode iterasi jacobian dan gauss-seidel dijamin konvergen apabila angka di diagonalnya lebih dominan dari yang lain dan tidak boleh nol (solusi: tukar baris)



* relaxation

ialah sedikit modifikasi untuk metode jacobian dan gauss-seidel, yaitu dipilih value lambda sehingga setiap terjadi iterasi, xinew tidak langsung menggantikan xiold, namun



Dimana nilai lambda dipilih diantara 0 dan 2, nilai lambda=1 adalah sama seperti no relaxation,

* underrelaxation (0<lambda<1)

dengan underrelaxation, xinew lebih ditentukan oleh xiold. Digunakan untuk membuat system yang tidak konvergen menajdi konvergen.

* Overrelaxation (1<lambda<2)

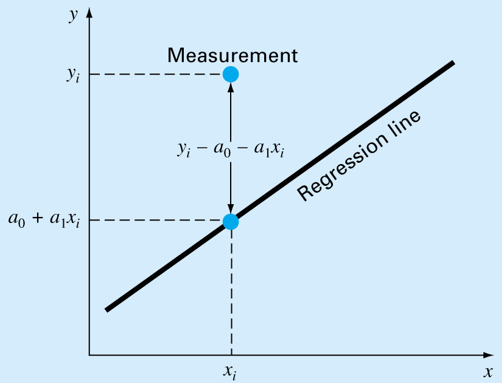
Dengan overrelaxation, proporsi xinew lebih kuat. Digunakan Ketika sistem konvergen namun terlalu lambat, penggunaan overrelaxation dapat mempercepat konvergensi

**CURVE FITTING**

**PART 1: REGRESSION**

Objektif: mencari garis lurus yg “terbaik” untuk merepresentasikan data

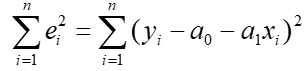
Residual (error) dari regression



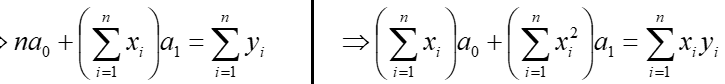
Beberapa metode mencari “best” fit

* Minimize sum of residuals (tdk valid, garis yg melewati tengah auto memenuhi)
* Minimize sum of absolute values of residuals (tdk valid, garis tdk unik)
* Minimax (bias Ketika bertemu outlier)

**Least-Square Fit**

****(sesuai statistik, solusi unik, mudah dihitung)

Dengan persdif diturunkan dgn a0 dan a1, didapat 2 persamaan:

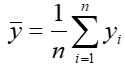
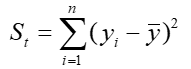
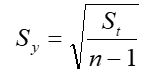


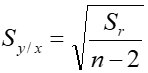
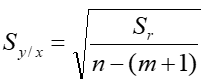
Penyelesaian persamaan tsb menghasilkan



(btw a1 adalah m, a0 adalah c)

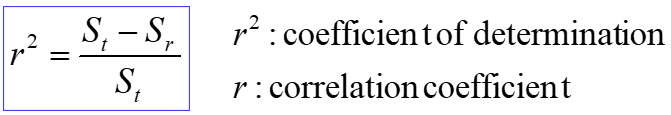
Simbol2 dalam persamaan regresi & it’s meaning:

*  ialah mean dari data
* ialah penjumlahan dari kuadrat deviasi (kek rumus varians tp gadibagi n-1)
*  standard deviation dari data
* ialah penjumlahan dari kuadrat residual
* Standard error of estimate

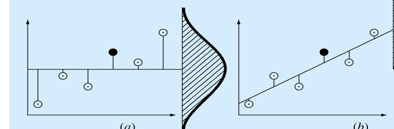
, atau dimana m adalah jumlah koef. a dlm persamaan

mirip dengan std. deviation, namun terhadap garis regresi / residual alih2 thdp mean

* Correlation coefficient



r^2 menandakan seberapa improve garis regresi untuk menggambarkan data daripada sekedar perhitungan mean.



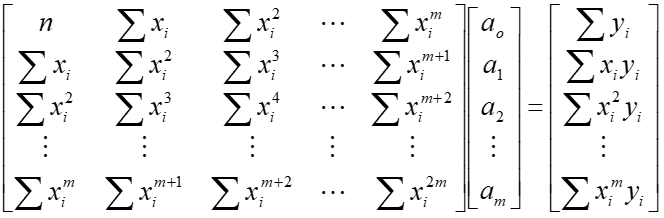
**Polynomial Regression**

Misal kumpulan titik (x1,y1),(x2,y2),...,(yn,yn)

Akan diregresikan dgn persamaan polinomial y=a0+ a1x + a2x2 + ... + amxm

Yg memiliki best fit terhadap titik2 data.

Maka koefisien a0,...,an dapat dicari dengan persamaan matriks:



**Multiple Linear Regression**

Ketika y adalah fungsi dari banyak variabel x1, x2, … , xm

Maka persamaan regresi akan berbentuk 

**General Linear Least Square**

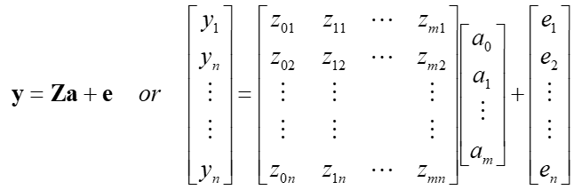
Misal ada sebuah garis f = 

Dan garis f dibuat untuk mendeskripsikan kumpulan data y1,…,yn. Maka untuk setiap y, dapat dideskripsikan dengan



Dimana y adalah titik, e adalah residu dari titik, dan z adalah basis function dari fungsi garis regresi f.

Notice bentuk tersebut, apabila dijalankan untuk **seluruh** titik (x1,y1),…(xn,yn), maka terbentuk matriks



* m adalah jumlah base function pd persamaan regresi linear
* n adalah jumlah titik^2 data
* y1,…yn adalah nilai-nilai y dari data
* z01, …, zij, …, zmn adalah nilai fungsi basis zi apabila dimasukkan xj

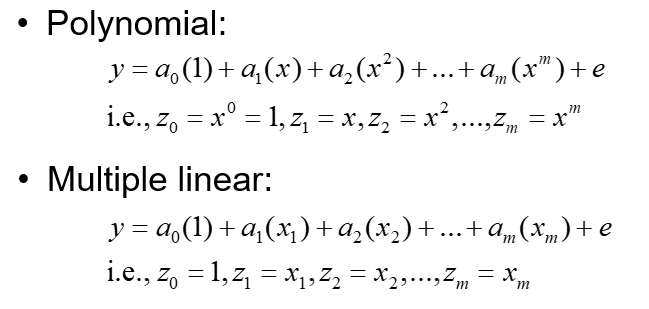
[ dengan kata lain, zij sama saja zi(xj) ]

dan untuk mencari nilai koefisien a0,…,am. Dicari ketika penjumlahan e^2 paling sedikit (least-square) yaitu dengan persamaan diferensial. Hasil akhir dari persamaan diferensial tsb menghasilkan persamaan matriks:



Sehingga {a} bisa dicari dan disubstitusi ke persamaan regresi 

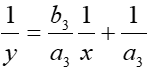
Note: bisa dilihat bahwa baik least square biasa, polynomial, dan multiple linear regression sebenarnya masih masuk dalam kategori general linear least square.



**Linearization of Nonlinear Relationships**

Beberapa fungsi dapat “dipaksakan” agar berbentuk linear dengan mengubah bentuk fungsi dan mendefinisikan ulang sumbu-x dan sumbu-y.

Contoh2:

* menjadi ; sumbu-y => ln(y)
*  menjadi  ; sumbu-y => log(y) ; sumbu-x => log(x)
*  menjadi  ; sumbu-y => 1/y ; sumbu-x => 1/x

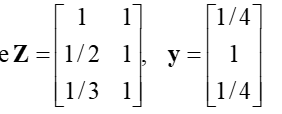
Kegunaan transformasi ini adalah dapat digunakan linear-regression untuk menggambarkan data dlm bentuk garis lurus. (dalam grafik, bentuknya menjadi seperti garis)

Algoritma:

1. Ubah bentuk fungsi menjadi “seperti linear”
2. Definisikan y’ dan x’ agar fungsi awal linear
3. Untuk setiap titik^2 fungsi, hitung nilai y’ dan x’ nya
4. Lakukan regresi linear untuk setiap titik (y’i, x’i)

Cth: diberikan data: (1,4), (2,1), (3,4) dimana y=a1x/(b1+x)

1. Ubah bentuk menjadi 1/y = (b1/a1)(1/x) + 1/a1
2. Definisikan y’=1/y ; x’=1/x dan persamaan menjadi y’=c1x’+c2
3. Data menjadi (1,1/4), (1/2,1), (1/3,1/4)
4. ZTZ{c}=ZT{y} dimana z1 = 1 dan z2 = x, sehingga



hitung dan dapatkan c1 & c2, sehingga best-fit ditemukan.