Optimization, fitting, and least-squares method

最適化、フィッティング、最小二乗法

神谷利夫

東京工業大学
科学技術創成研究院
フロンティア材料研究所

東工大講義資料 / Tokyo Tech Lecture Materials (English+Japanese)
http://conf.msl.titech.ac.jp/Lecture/python/index-numericalanalysis.html
pythonによる最小二乗法・最適化問題 GUIプログラミング (Japanese)
LSQ/Optimization GUI programing (Japanese)
http://conf.msl.titech.ac.jp/Lecture/python/tutorial-optimize/index-python-optimize-

Linear lest squares method (*l*LSQ)

線形最小自乗法

Approximation of many sample points: Minimization (Optimization)

(多数の標本点の近似: 最小化問題)

How to determine most plausible parameters *a* and *b*
if observed data (x_1, y_1), ・・・ (x_n, y_n) follow $f(x) = a + bx$,
※ Error ε_i should be considered: $y_i = f(x_i) + \varepsilon_i$

Fundamental idea: Determine *a* and *b* so as to minimize (maximize) a target function *S* (e.g., error residual function (残差関数))

Minimax method (ミニマックス法) : $S = \Sigma|f(x_i) - y_i|$
Least-squares (LSQ) method (最小自乗法): $S = \Sigma(f(x_i) - y_i)^2$

$$S = \Sigma(a + bx_i - y_i)^2$$
$$dS/da = 2\Sigma(a + bx_i - y_i) = 2an + 2b\Sigma x_i - 2\Sigma y_i = 0$$
$$dS/db = 2\Sigma x_i(a + bx_i - y_i) = 2a\Sigma x_i + 2b\Sigma x_i^2 - 2\Sigma x_i y_i = 0$$

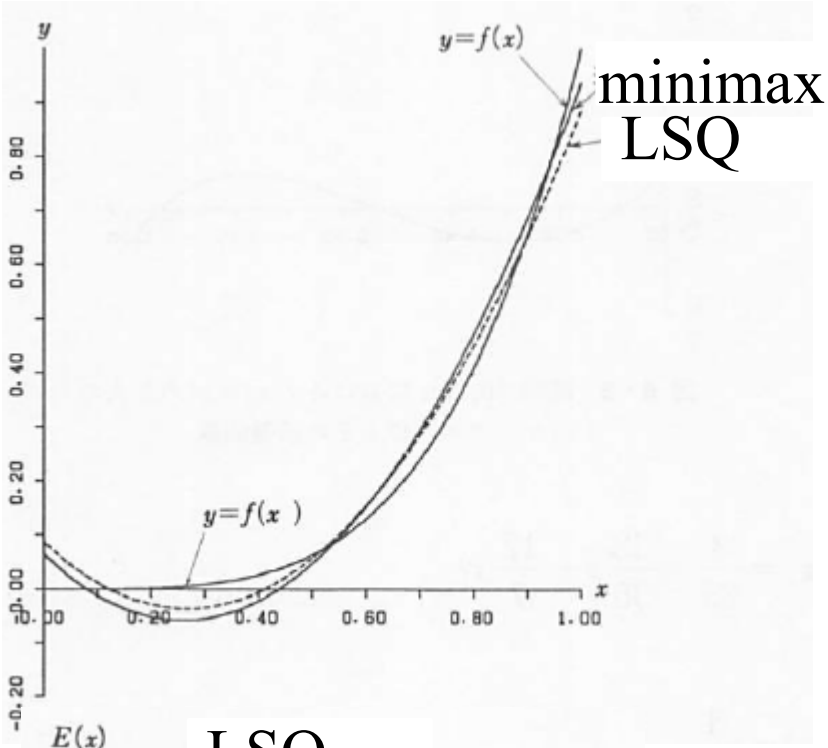
$$\begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix}$$

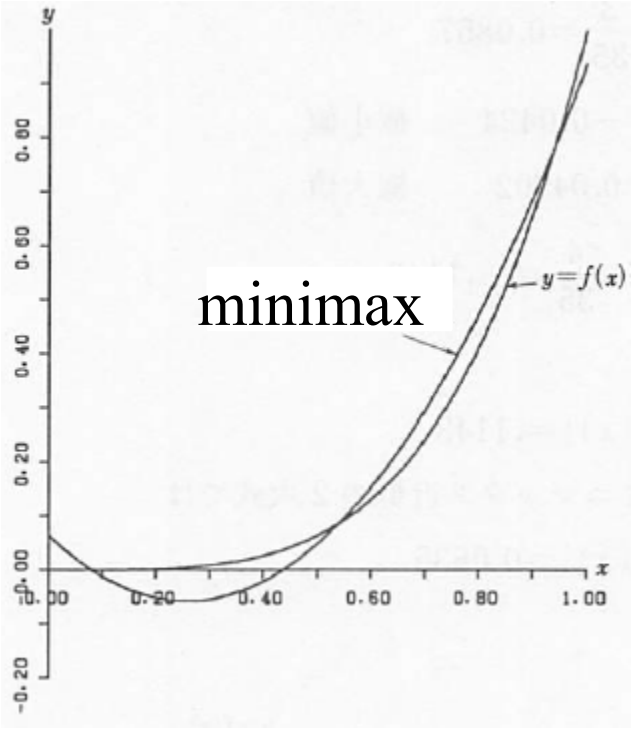
Even for $f(x) = a + bx + cx^2 +$・・・, only one matrix operation can give a final solution

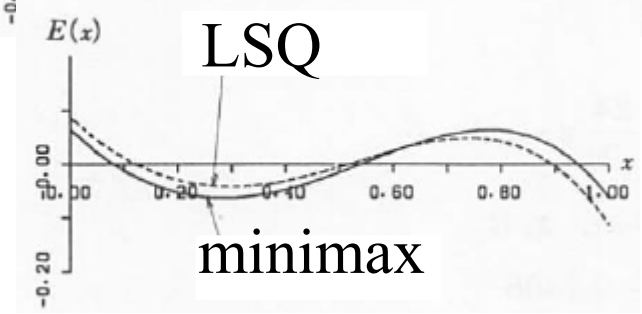
Minimax approximation (ミニマックス近似)

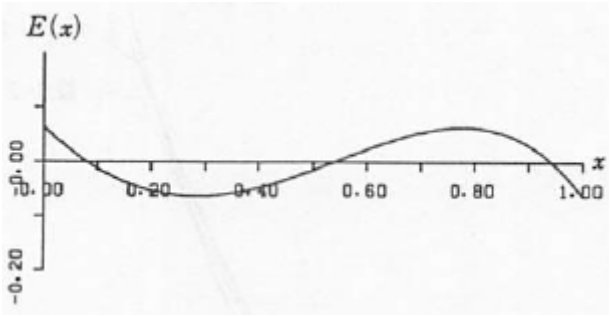
入門 数値計算

Minimize $\max_{a \le x \le b} |g(x) - f(x)|$









*l*LSQ: Polynomial

線形最小二乗法: 多項式

$$f(x)=\sum_{k=0}^{n} a_k x^k \qquad S=\sum_{i=1}^{N}\left(y_i-\sum_{k=0}^{n} a_k x_i^{\,k}\right)^2$$

$$\frac{dS}{da_l}=-2\sum_{i=1}^{N} x_i^{\,l}\left(y_i-\sum_{k=0}^{n} a_k x_i^{\,k}\right)=0$$

$$\sum_{k=0}^{n}\sum_{i=1}^{N} a_k x_i^{\,k+l}=\sum_{i=1}^{N} y_i x_i^{\,l} \qquad (l=0,\,1,\cdots,N)$$

$$\begin{pmatrix} n & \sum x_i & \sum x_i^{\,2} & \cdots & \sum x_i^{\,N} \\ \sum x_i & \sum x_i^{\,2} & \sum x_i^{\,3} & & \sum x_i^{\,N+1} \\ \sum x_i^{\,2} & \sum x_i^{\,3} & \sum x_i^{\,4} & & \sum x_i^{\,N+2} \\ \vdots & & & \ddots & \\ \sum x_i^{\,N} & \sum x_i^{\,N+1} & \sum x_i^{\,N+2} & & \sum x_i^{\,2N} \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}=\begin{pmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^{\,2} \\ \vdots \\ \sum y_i x_i^{\,N} \end{pmatrix}$$

*l*LSQ: General functions

線形最小二乗法: 一般関数の場合

$$f(x)=\sum_{k=1}^{n} a_k f_k(x) \qquad S=\sum_{i=1}^{N}\left(y_i-\sum_{k=1}^{n} a_k f_k(x_i)\right)^2$$

$$\frac{dS}{da_l}=-2\sum_{i=1}^{N} f_l(x_i)\left(y_i-\sum_{k=1}^{n} a_k f_k(x_i)\right)=0$$

$$\begin{pmatrix} \sum f_1(x_i)f_1(x_i) & \sum f_1(x_i)f_2(x_i) & \sum f_1(x_i)f_3(x_i) & \cdots & \sum f_1(x_i)f_N(x_i) \\ \sum f_2(x_i)f_1(x_i) & \sum f_2(x_i)f_2(x_i) & \sum f_2(x_i)f_3(x_i) & & \sum f_2(x_i)f_N(x_i) \\ \sum f_3(x_i)f_1(x_i) & \sum f_3(x_i)f_2(x_i) & \sum f_3(x_i)f_3(x_i) & & \sum f_3(x_i)f_N(x_i) \\ \vdots & & & \ddots & \\ \sum f_N(x_i)f_1(x_i) & \sum f_N(x_i)f_2(x_i) & \sum f_N(x_i)f_3(x_i) & & \sum f_N(x_i)f_N(x_i) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} \sum y_i f_1(x_i) \\ \sum y_i f_2(x_i) \\ \sum y_i f_3(x_i) \\ \vdots \\ \sum y_i f_N(x_i) \end{pmatrix}$$

If *f(x)* is *linear with respect to fitting parameters,*
final solution is obtained by one matrix operation

係数に関して線形であれば、1度の行列計算で最終解が得られる

ex. $f(x)=a+b\log x+c/x$

$$f(x,y)=a+bxy+cy/x$$

Ex of *l*LSQ: Lattice spacing of triclinic lattice

(三斜晶結晶の面間隔)

$$d_{hkl}^{-2} = \left|\mathbf{G}_{hkl}\right|^2 = \left|h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^*\right|^2$$

$$\frac{1}{d_{hkl}^{\ 2}} = S_{11}h^2 + S_{22}k^2 + S_{33}l^2 + 2S_{12}hk + 2S_{23}kl + 2S_{31}lh$$

$$S_{11} = \mathbf{a}^* \cdot \mathbf{a}^* = b^2c^2\sin^2\alpha / V^2$$

$$S_{22} = c^2a^2\sin^2\beta / V^2$$

$$S_{33} = c^2a^2\sin^2\gamma / V^2$$

$$S_{12} = \mathbf{a}^* \cdot \mathbf{b}^* = abc^2\left(\cos\alpha\cos\beta - \cos\gamma\right)/V^2$$

$$S_{23} = a^2bc\left(\cos\beta\cos\gamma - \cos\alpha\right)/V^2$$

$$S_{31} = ab^2c\left(\cos\gamma\cos\alpha - \cos\beta\right)/V^2$$

$$V = abc\sqrt{1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma + 2\cos\alpha\cos\beta\cos\gamma}$$

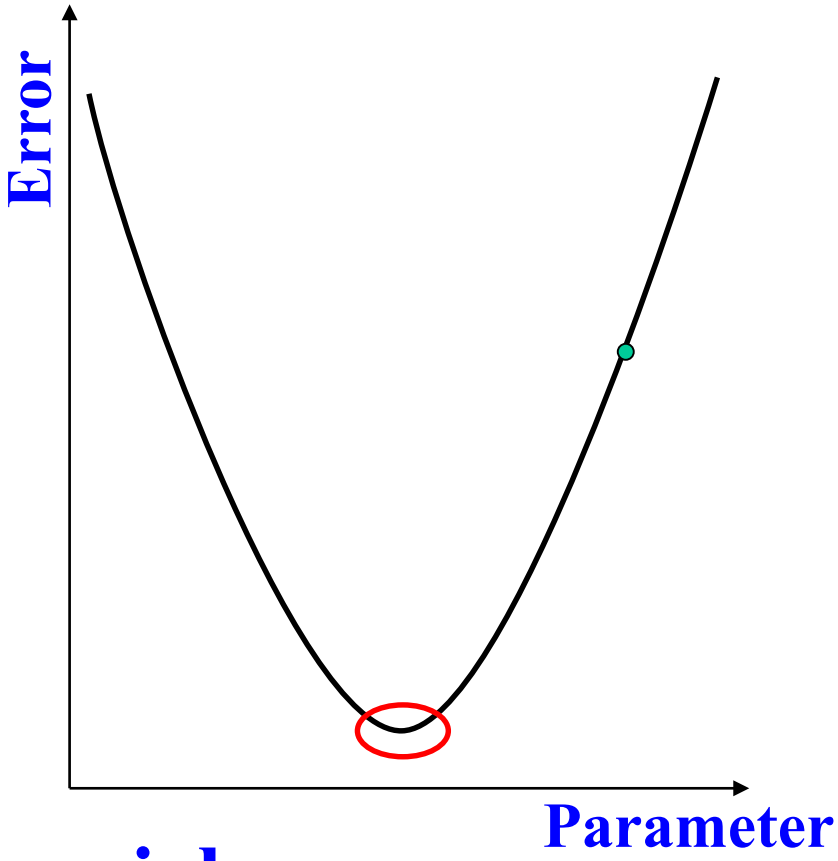
The form of d_{hkl}^{-2} is a linear function with respect to S_{ij}.

1. S_{ij} is obtained by *l*LSQ
2. S_{ij} => Reciprocal lattice parameters (a^*, b^*, c^*, α^*, β^*, γ^*)
3. => Lattice parameters (a, b, c, α, β, γ)

Self-consistent method
自己無撞着法

Least-squares method

- Fitting to a function having linear parameters
- Final solution is obtained just by one matrix operation
- Unique solution



Four or higher order polynomial, Transcendental equation (超越方程式)

- Difficult to have an analytical solution
- Even numerical analysis cannot give final solution by one-cycle calculation

＝＞ **Iterative calculation** (反復計算)

Example of simple self-consistent (SC) calc

A simple case: Solve $g(x) = 0$

SC method is applicable by converting to $x = g(x) + x = f(x)$

Note: not efficient nor stable for many cases

Simple procedure:

Initial value x_0

1st iteration : $x_1 = f(x_0)$

2nd iteration: $x_2 = f(x_1)$ ….

Difficultt to converge: Diverge, Oscillation

(収束しにくい：発散、振動)

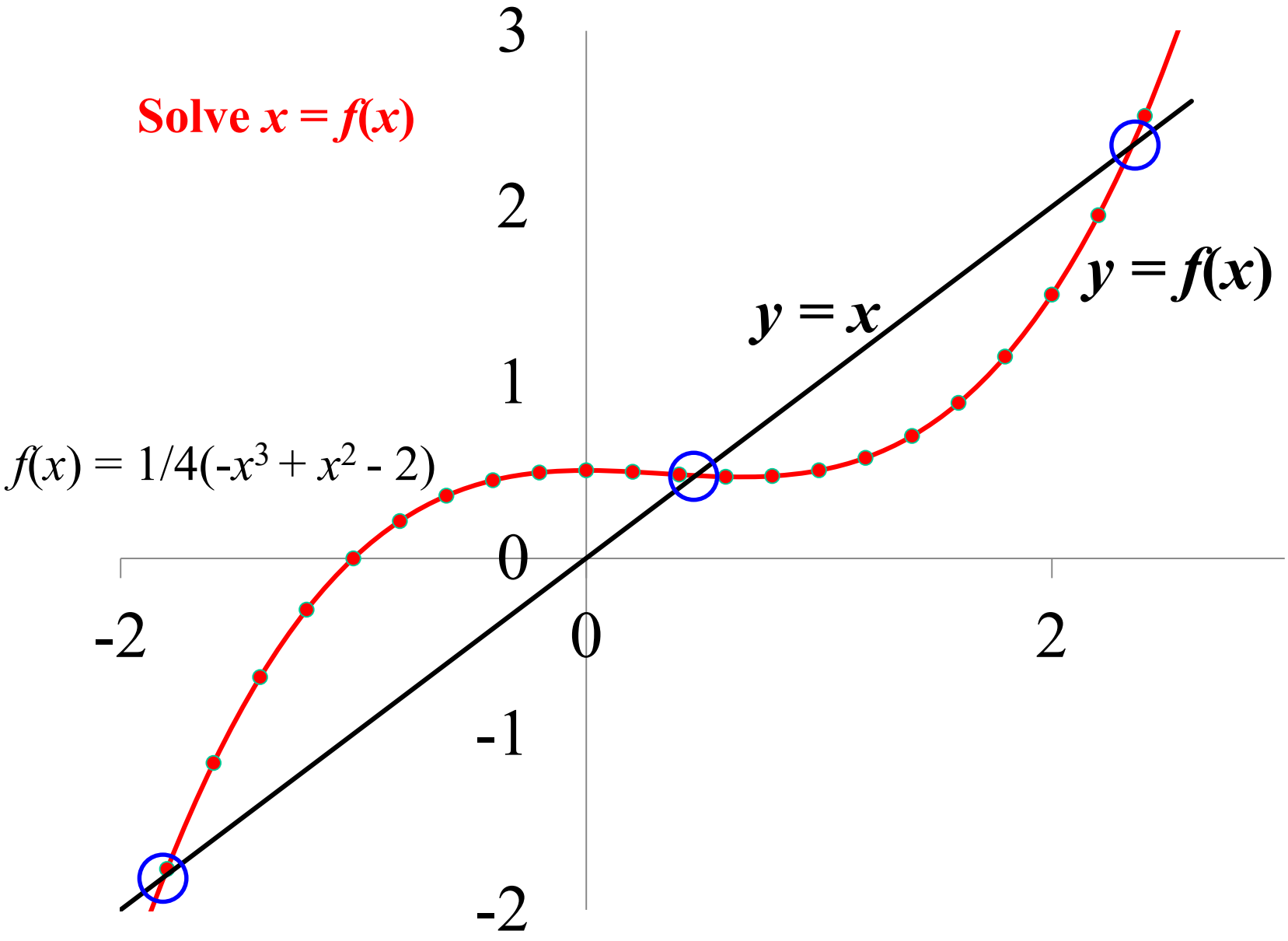
Mixing factor (混合係数) **k_{mix}: Stabilize convergence**

Initial value x_0

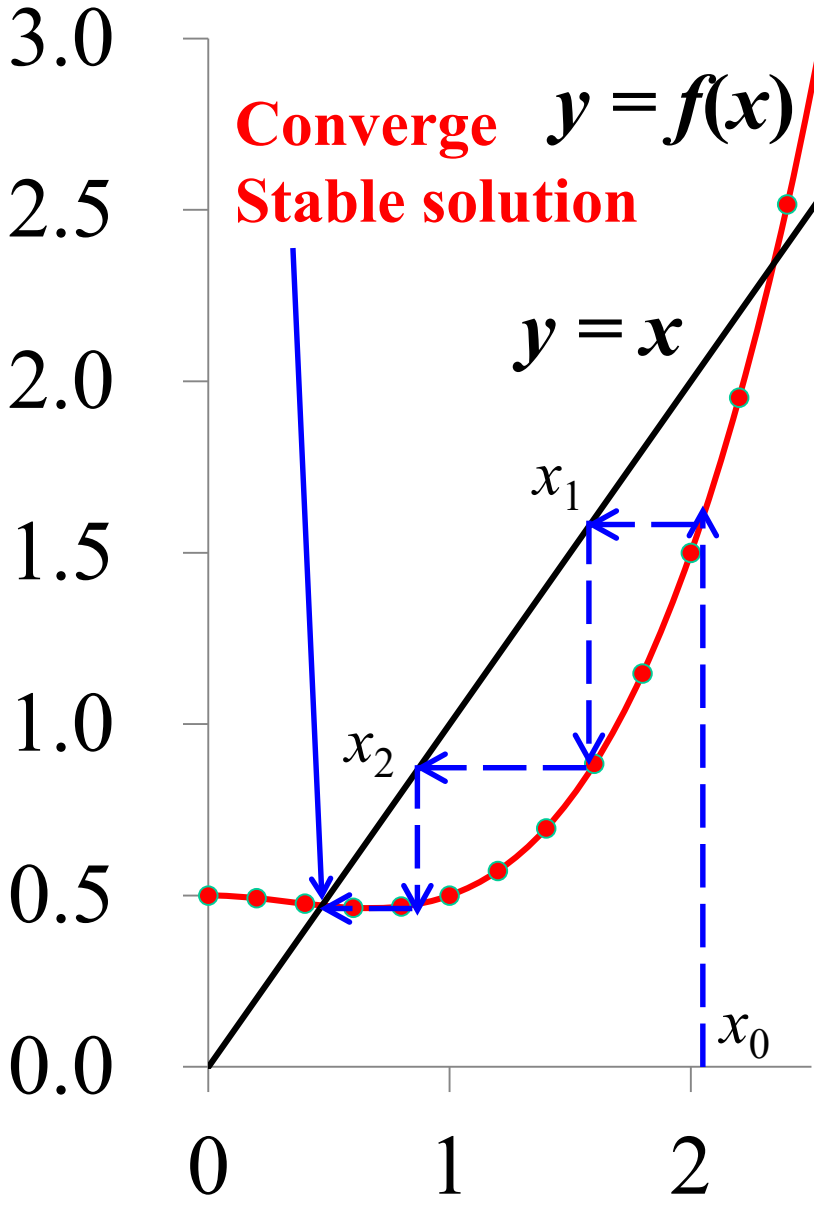
1st iteration : $x_1 = f(x_0)$ => $x_1' = (1 - k_{mix})\, x_0 + k_{mix} x_1$

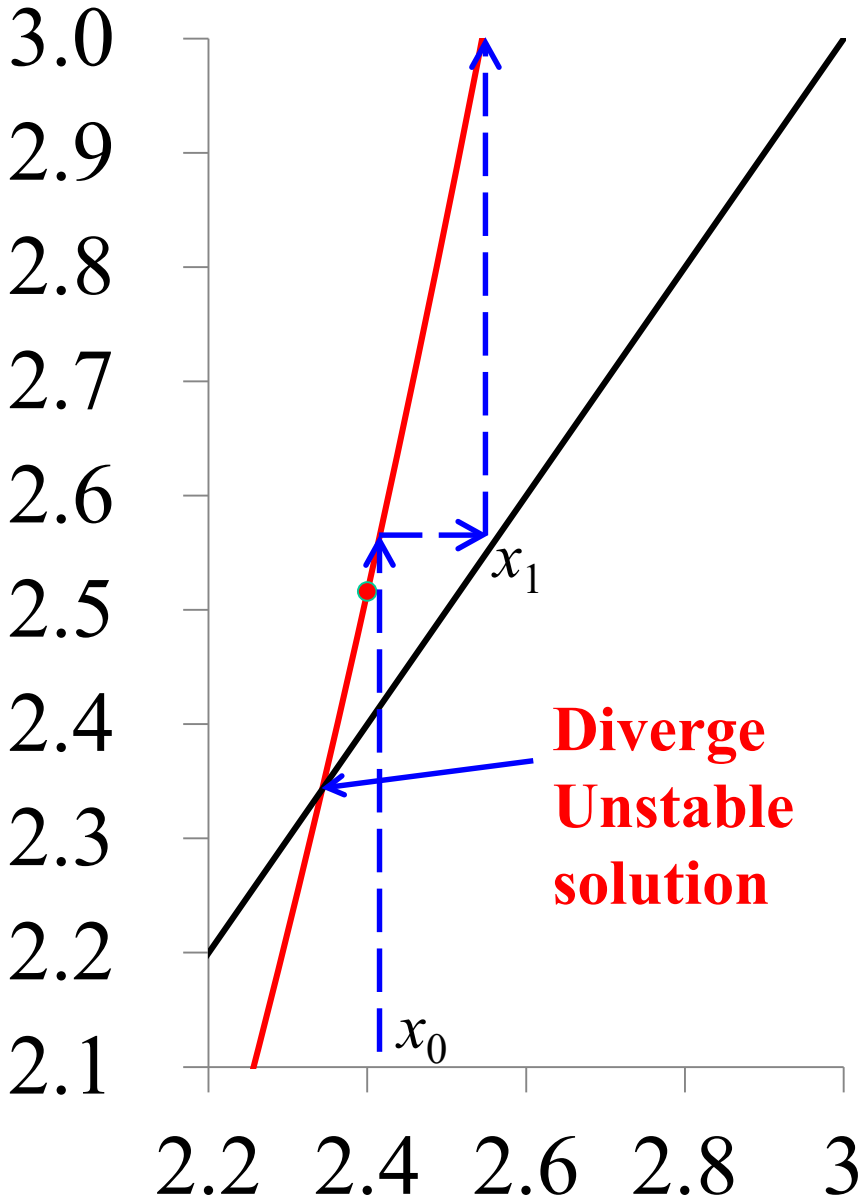
2nd iteration: $x_2 = f(x_1')$ ….

Illustrative explanation of SC



SC: Convergence process





$f'(x) < 1$ must be satisfied for convergence

Example of SC: Diode with series resistance

$$I = I_0\left[\exp\left(\frac{e}{nkT}(V - RI)\right) - 1\right]$$

Repeat

$$I_i = I_0\left[\exp\left(\frac{e}{nkT}(V - RI_{i-1})\right) - 1\right]$$

until abs($I_i = I_{i-1}$) < EPS is achieved

- E.g., initial voltages would be chosen as $V/2$ for the diode and the R
- This SC is not so stable; mixing factor k should be adjusted

For sequential calculations of $I - V$ characteristic, e.g., V from 0.0 to 1.0, using a preconverged result for the initial value of the next V will enhance convergence.

例えばVを順次変えてI-V特性を計算するような場合、すでに収束した値を次のVにおける初期値として利用すると早く収束できる。

i	I	Ical	|error|	I0=	1.E-12	A
0	2	-1E-12	2	n=	1	
1	1.8	-1E-12	1.8	T=	300	K
2	1.62	-1E-12	1.62	R=	1	ohm
3	1.458	-1E-12	1.458	V=	1	
4	1.3122	-1E-12	1.3122			
5	1.18098	-1E-12	1.18098	k=	0.1	
6	1.062882	-9.1E-13	1.06288			
7	0.956594	4.31E-12	0.95659			
8	0.860934	2.09E-10	0.86093			
9	0.774841	5.77E-09	0.77484			
10	0.697357	1.14E-07	0.69736			
11	0.627621	1.66E-06	0.62762			
12	0.564859	1.86E-05	0.56484			
13	0.508375	0.000163	0.50821			
14	0.457554	0.00115	0.4564			
15	0.411914	0.006655	0.40526			
16	0.371388	0.031631	0.33976			
17	0.337412	0.116849	0.22056			
18	0.315356	0.272927	0.04243			
19	0.311113	0.321305	0.01019			
20	0.312132	0.308953	0.00318			
21	0.311814	0.312754	0.00094			
22	0.311908	0.311626	0.00028			
23	0.31188	0.311965	8.5E-05			
24	0.311888	0.311863	2.5E-05			
25	0.311886	0.311893	7.6E-06			
26	0.311887	0.311884	2.3E-06			

First-principles calculation :

Self-consistent field (SCF,自己無撞着) calculation

- Hamiltonian of one-electron quantum equation includes wave functions

$$\left\{-\frac{1}{2}\nabla_l - \sum_m \frac{Z_m}{r_{lm}} + \sum_m \int \frac{\rho_m(\mathbf{r}_m)}{r_{lm}} d\mathbf{r}_m + V_{Xl}(\mathbf{r}_l)\right\}\phi_l(\mathbf{r}_l) = \varepsilon_l \phi_l(\mathbf{r}_l)$$

- First-step calculation requires electron density guessed / assumed ρ_{ini} :
  e.g,. by uniform density, sum of atomic electron density,,,

- Electron density ρ_{fin} is calculated the solved wave functions, but ρ_{fin} would be different from ρ_{ini}

ρ_{ini} must be equal to ρ_{fin}, otherwise these loss physical meaning

- More appropriate ρ_{new} is guessed from ρ_{fin} and ρ_{ini}, and repete the above calculations

  ex. : $\rho_{new} = \rho_{ini} + k_{mix}(\rho_{fin} + \rho_{ini})$

  k_{mix} : Mixing factor

  A parameter to suppress divergence of the SCF calculation
  close to 1 would be easily diverged, close to 0 causes slow convergence

SCF cycyle
Repeat until $\rho_{fin} = \rho_{ini}$

Example: SCF/structure relaxation by VASP

tkamiya@csrv0:~/Work/LaCrAsO/SpinPolarized

ファイル(F)　編集(E)　表示(V)　端末(T)　タブ(B)　ヘルプ(H)

```
  1 F= -.24922201E+03 E0= -.24922201E+03  d E =-.249222E+03  mag=    17.6753
 curvature:   0.00 expect dE= 0.000E+00 dE for cont linesearch  0.000E+00
 trial: gam= 0.00000 g(F)=  0.620E+00 g(S)=  0.305E-01 ort = 0.000E+00 (trialstep = 0.100E+01
)
 search vector abs. value=  0.650E+00
 bond charge predicted
       N       F                      dE             d eps       ncg     rms          rms(c)
DAV:   1    -0.249256423264E+03   -0.24926E+03   -0.54781E+01  3528   0.200E+01    0.196E+00
DAV:   2    -0.249670978228E+03   -0.41455E+00   -0.52988E+00  4416   0.955E+00    0.161E+00
DAV:   3    -0.249672461360E+03   -0.14831E-02   -0.53814E-01  4640   0.336E+00    0.153E+00
DAV:   4    -0.249667045995E+03    0.54154E-02   -0.45192E-01  4632   0.183E+00    0.129E+00
DAV:   5    -0.249662986402E+03    0.40596E-02   -0.16171E-01  4664   0.134E+00    0.113E+00
DAV:   6    -0.249664501455E+03   -0.15151E-02   -0.86520E-02  4520   0.152E+00    0.943E-01
DAV:   7    -0.249658663938E+03    0.58375E-02   -0.36669E-02  4626   0.103E+00    0.315E-01
DAV:   8    -0.249657255947E+03    0.14080E-02   -0.11030E-02  4432   0.529E-01    0.406E-01
DAV:   9    -0.249656661683E+03    0.59426E-03   -0.64937E-03  3424   0.480E-01    0.219E-01
DAV:  10    -0.249654538004E+03    0.21237E-02   -0.11755E-03  2528   0.225E-01    0.151E-01
DAV:  11    -0.249654612437E+03   -0.74432E-04   -0.11566E-03  2520   0.213E-01
   2 F= -.24965461E+03 E0= -.24965461E+03  d E =-.432599E+00  mag=    18.2912
 trial-energy change:   -0.432599  1 .order   -0.416777   -0.650072   -0.183481
 step:   1.3105(harm=  1.3932)  dis= 0.06748  next Energy=  -249.683568 (dE=-0.462E+00)
 bond charge predicted
       N       F                      dE             d eps       ncg     rms          rms(c)
DAV:   1    -0.249658788237E+03   -0.24966E+03   -0.53760E+00  3536   0.623E+00    0.599E-01
DAV:   2    -0.249698102900E+03   -0.39315E-01   -0.48908E-01  4528   0.303E+00    0.671E-01
```

Typical iteration of SC calculation

Find the solution of $f(x, \rho(x)) = 0$:

Case this is easily done if $\rho(x)$ is provided

1. Assum $\rho(x)$ and solve $f(x, \rho(x)) = 0$ to get approximate x_i
2. Calculate $\rho(x_i)$ with the obtained x_i, solve $f(x, \rho(x_i)) = 0$, and get improved approximation x_{i+1}
3. Rpeat 1-2 so as to decrease $|\rho(x_{i+1}) - \rho(x_i)|$, $|x_{i+1} - x_i|$ to required accuracy

Self-consistent approach (自己無動着計算)

May be diverged if the obtained x_i' is used for x_{i+1}

=> Stabilize converged using mixing factor (混合係数) k_{mix}

Initial x_0

First iteration: $x_1 = f(x_0)$ => $x_1' = (1 - k_{mix})\, x_0 + k_{mix} x_1$

Next iteration: $x_2 = f(x_1')$ ….

Problems of SC calculations

- **Some solutions would not be obtained** (収束しない解があり得る)

  $f'(x) < 1$ must be satisfied at the solution
  to obtain the solution of $x = f(x)$
  => Conversion of the equation may help, but not always

- **Convergence is not stable**

  mixing factor may improve

For many cases, use another method such as Newton method

- **Cases SC method is effective**

  Initial values close to the solution
  Effect of SC parameters is small to the equation
  (自己無撞着変数の方程式への影響が小さい)
  SC parameters have good convergence
  (自己無撞着変数の収束特性が良く、予測できる場合)

Transcendental equation

超越方程式の解法

Newton-Raphson method

Solve $f(x) = 0$

$$f(x_0+dx) = f(x_0) + dx\, f'(x_0) \sim 0$$
$$\Rightarrow \quad x_1 = x_0 + dx = x_0 - f(x_0) / f'(x_0)$$

$f'(x_0)$ can be substituted with finite diffrence

Secant method (割線法, はさみうち法)**:**

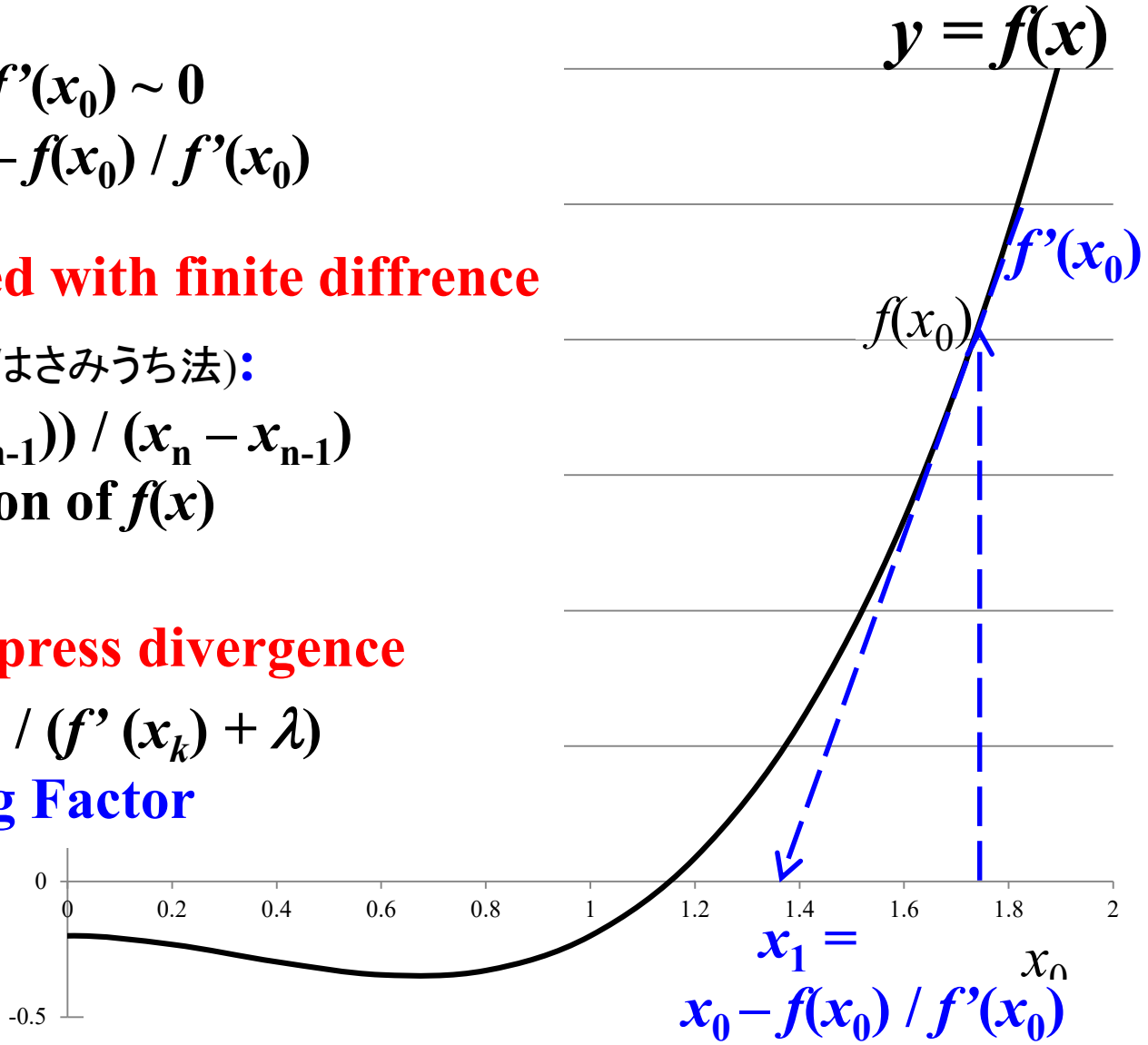
$$f'(x) = (f(x_n) - f(x_{n-1})) / (x_n - x_{n-1})$$

Save calculation of $f(x)$

Variation to suppress divergence

$$x_{k+1} = x_k - f(x_k) / (f'(x_k) + \lambda)$$

λ: Dumping Factor



Effect of dumping factor (収束過程の比較)

$f(x) = \exp(x) - 3x = 0$ (initial $x = 0$) Exact 0.619061

Newton-Raphson (Dumping factor = 0)

1	0.5	
2	0.610059654958962	0.110059654958962
3	0.61899677974154	0.00893712478257794
4	0.619061283355313	6.4503613773092e-005
5	0.619061286735945	3.38063244722622e-009
6	0.619061286735945	-1.94296000199483e-016

Newton-Raphson (Dumping factor = 0.1)

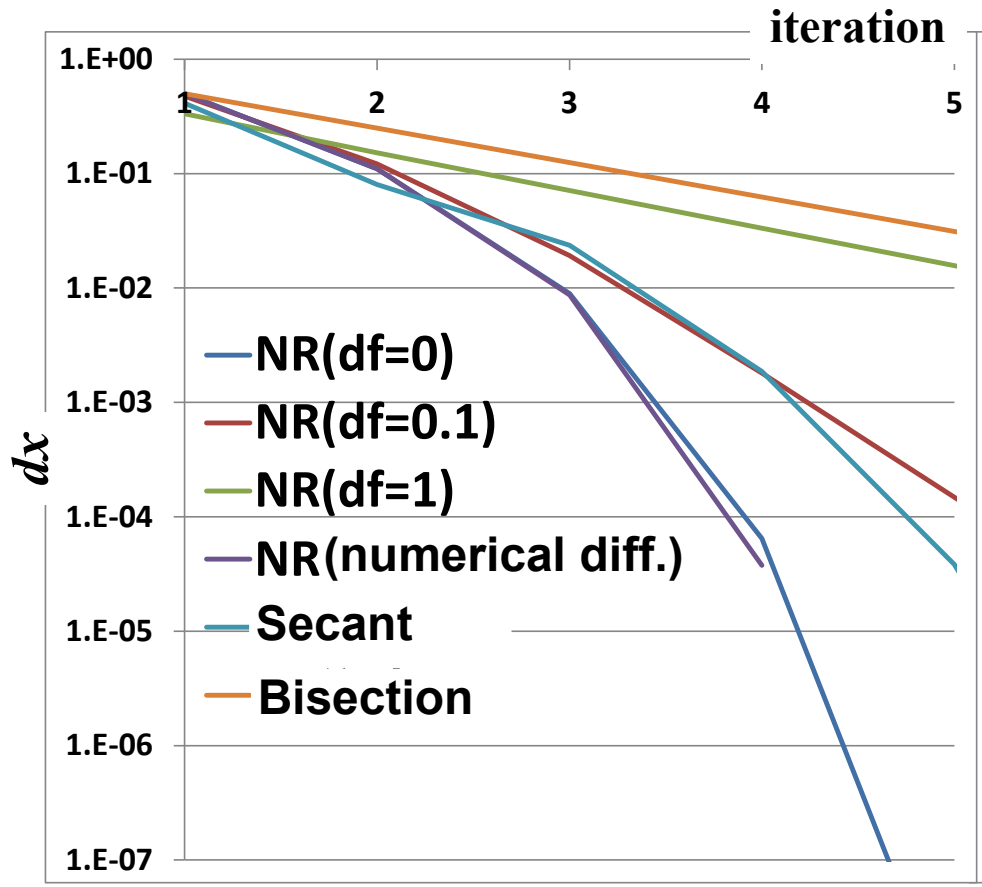
1	0.476190476190476	
2	0.597901649246081	0.121711173055605
3	0.617090542717403	0.0191888934713221
4	0.618900291486661	0.00180974876925825
5	0.619048316423879	0.000148024937217564
6	0.619060243007723	1.19265838440254e-005
7	0.619061202754359	9.59746635487409e-007
8	0.619061279978579	7.72242198569211e-008
9	0.619061286192231	6.21365241490959e-009
10	0.619061286692197	4.99965669237101e-010
11	0.619061286732425	4.0228535713285e-011

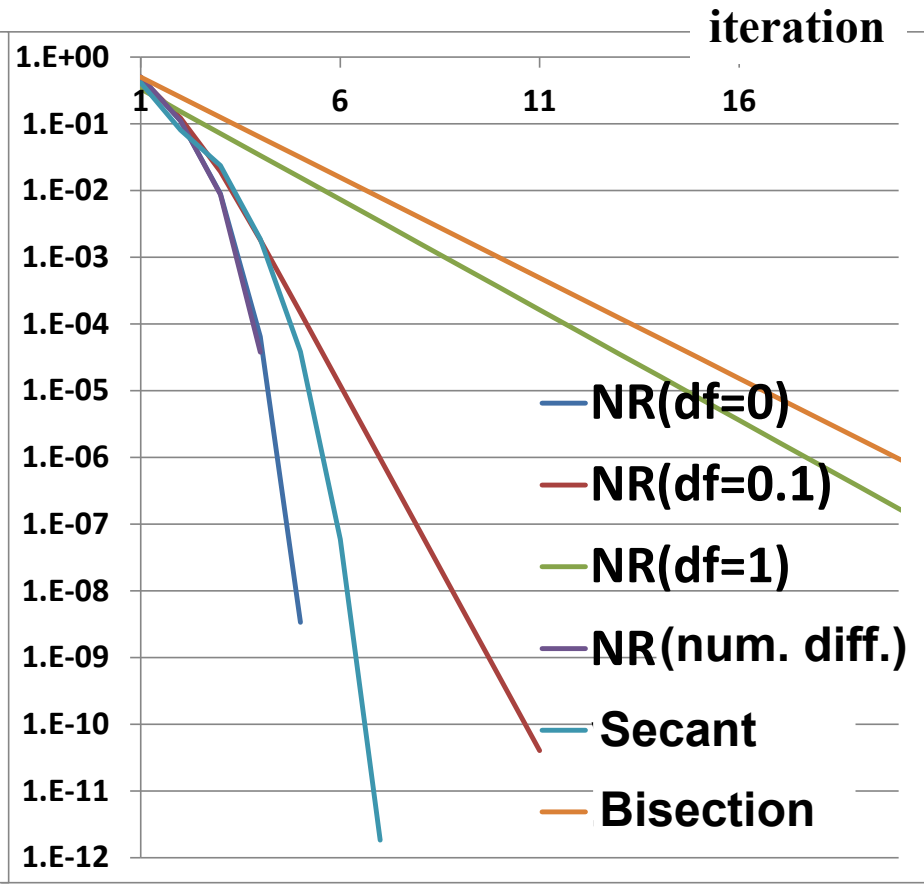
Newton-Raphson (Dumping factor = 1.0)

1	0.333333333333333	
2	0.485235618882813	0.15190228554948
3	0.556317491275292	0.0710818723924794
4	0.589692022113926	0.0333745308386341
5	0.605333177012923	0.0156411548989961
6	0.612649553494255	0.00731637648133212
7	0.616067929129785	0.00341837563553035
8	0.617664103982484	0.00159617485269905
9	0.618409199563502	0.00074509558101794
10	0.618756961315507	0.000347761752005284
11	0.618919262817103	0.000162301501596124
12	0.618995007056658	7.57442395542543e-005

Effect of dumping factor: Convergence process

$f(x) = \exp(x) - 3x = 0$ (initial $x = 0$) Exact 0.619061



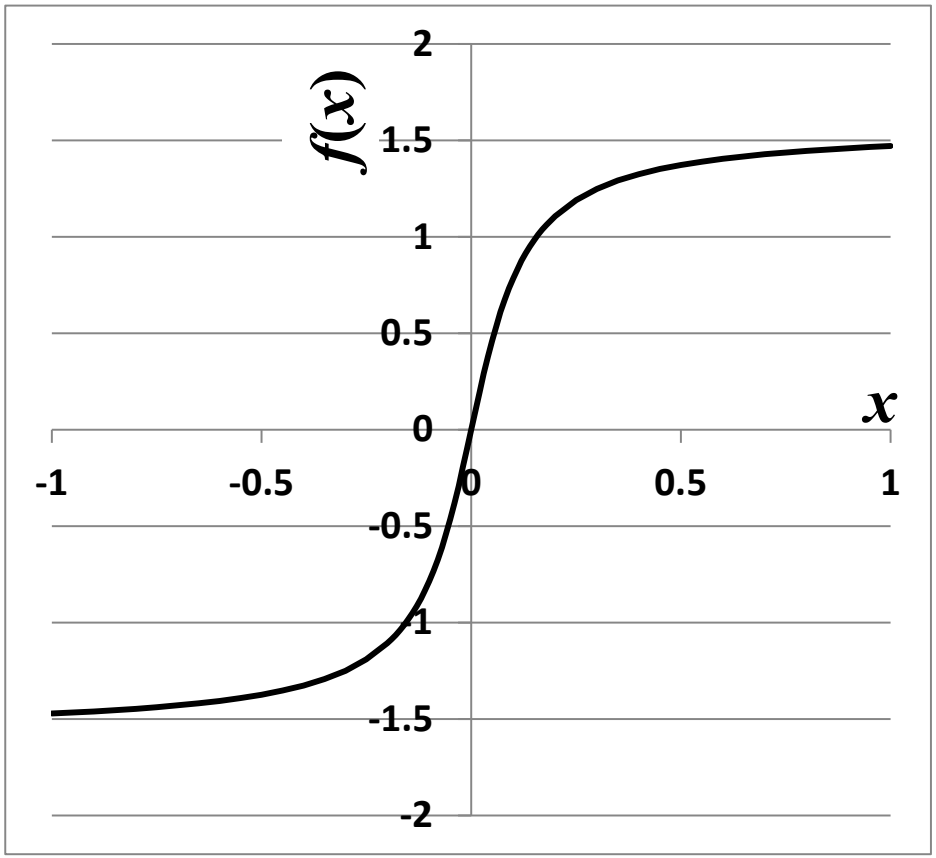


NR: Newton-Raphson method
df: Dumping Factor

Case Newton method fails

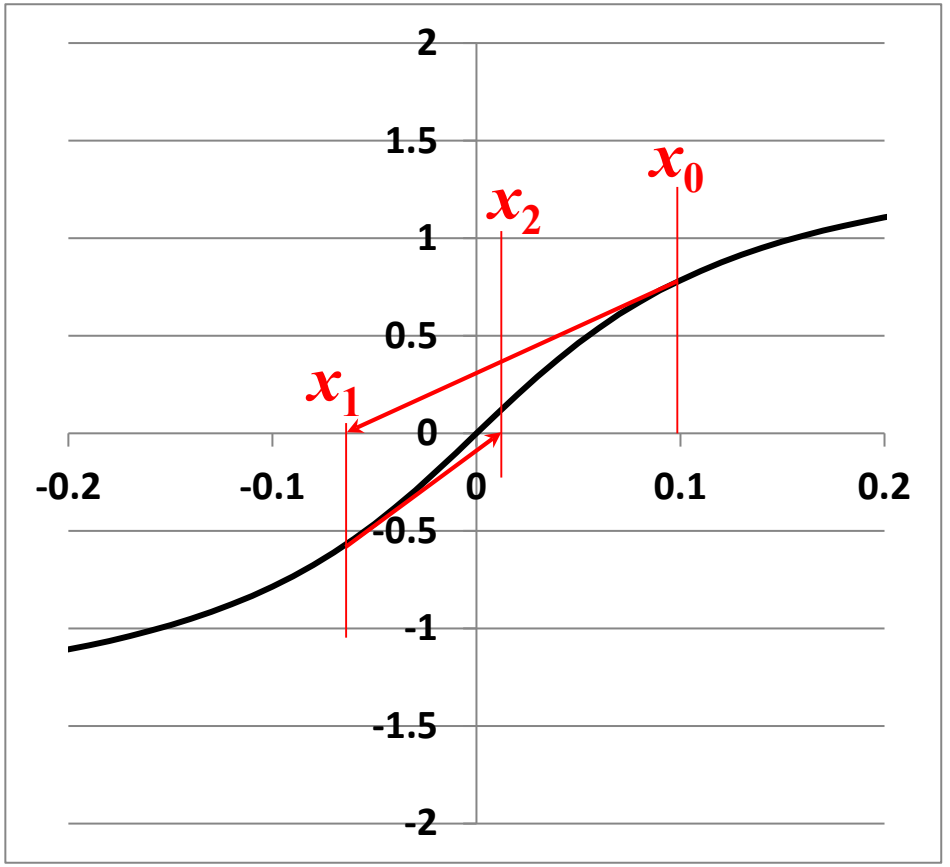
$f(x) = \tan^{-1}(10x)$

initial $x = 0.1$



Case for convergence

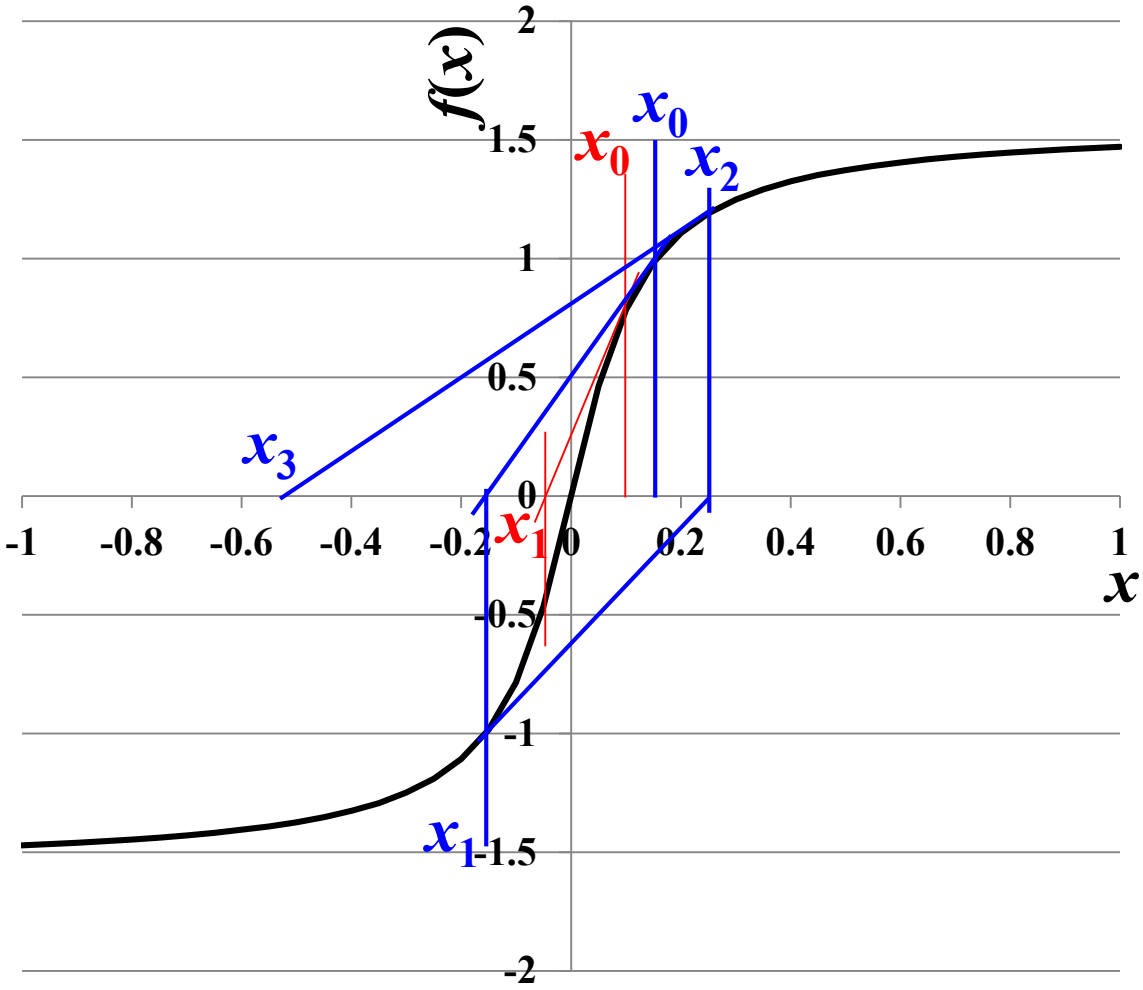
i	x	f(x)	df/dx	dx
0	0.1	0.7854	5	−0.1571
1	−0.05708	−0.5187	7.54257	0.06877
2	0.011686	0.11633	9.86527	−0.0118
3	−0.00011	−0.0011	9.99999	0.00011
4	1.15E−10	1.2E−09	10	−1E−10



Case Newton method fails

$f(x) = \tan^{-1}(10x)$

initial $x = 0.15$



Diverged ($\lambda = 0$)

i	x	f(x)	df/dx	dx
0	0.15	0.98279	3.07692	−0.3194
1	−0.16941	−1.0375	2.58404	0.40152
2	0.232112	1.164	1.56553	−0.7435
3	−0.51141	−1.3777	0.36827	3.74095
4	3.229546	1.53984	0.00958	−160.76
5	−157.529	−1.5702	4E−06	389644
6	389486.7	1.5708	1.1E−12	−1E+12

$$x_{k+1} = x_k - f(x_k) / (f'(x_k) + \lambda)$$

λ: Dumping Factor

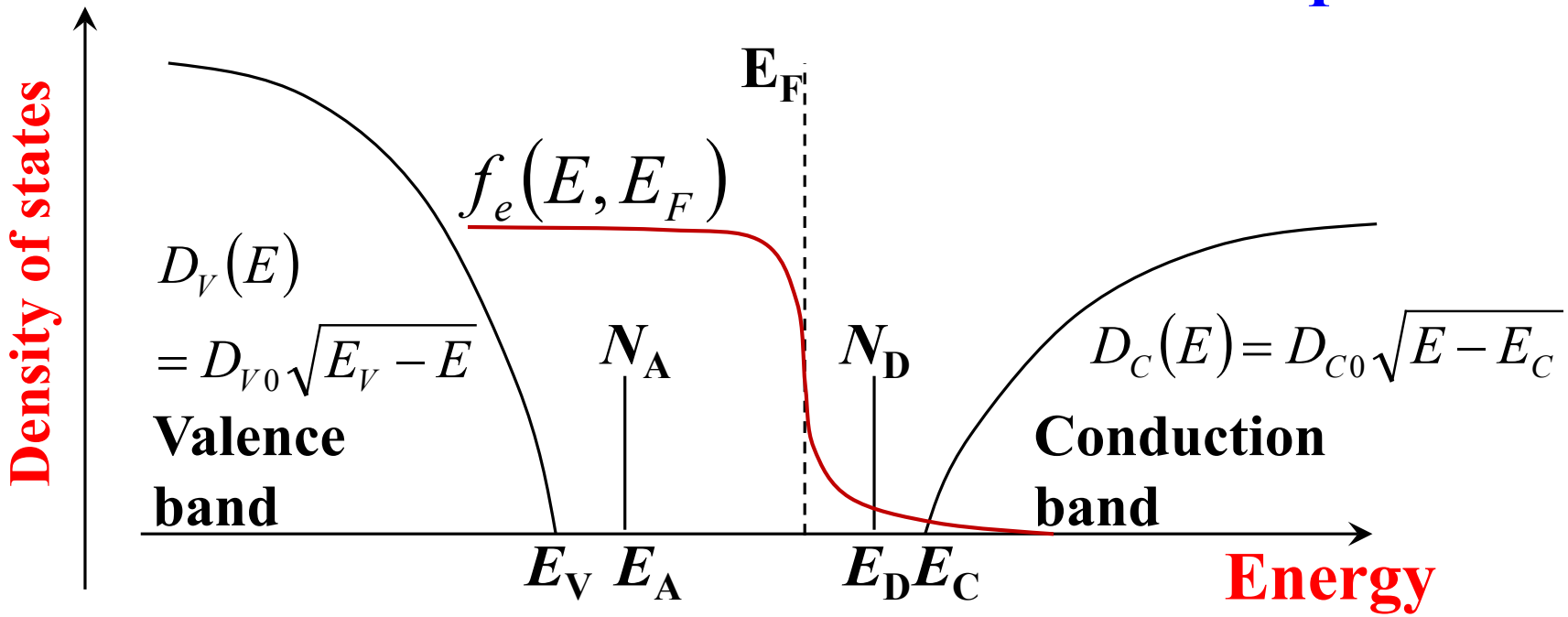
Stabilize convergence by choosing $\lambda (\lambda = 1)$

i	x	f(x)	df/dx	dx
0	0.15	0.98279	3.07692	−0.2411
1	−0.09106	−0.7387	5.46675	0.11422
2	0.023161	0.2276	9.49088	−0.0217
3	0.001466	0.01466	9.99785	−0.0013
4	0.000133	0.00133	9.99998	−0.0001
5	1.21E−05	0.00012	10	−1E−05
6	1.1E−06	1.1E−05	10	−1E−06
7	1E−07	1E−06	10	−9E−08
8	9.09E−09	9.1E−08	10	−8E−09
9	8.27E−10	8.3E−09	10	−8E−10

Semiconductor statistics (半導体統計): Calc. procedure under equilibrium

1. Determine parameters (*e.g.*, m_e^*) and related constants (N_c, D_{c0} etc)
2. Calculate density-of-states (DOS) function, $D(E)$
3. Consider '**Charge Neutrality Condition** (電荷中性条件)' at 0 K
4. At thermal equilibrium, E_F is independent of position.
   Draw a band diagram and obtain the energy levels of conduction band minimum (CBM) and valence band maximum (VBM), $E_C(x)$ & $E_V(x)$
5. Calculate extra charges $\rho_e(x)$ and $\rho_h(x)$ by
   $$\rho_e(x) = N_c \exp(-(E_{CBM}(x) - E_F) / k_B T)$$
   $$\rho_h(x) = N_v \exp(-(E_F - E_{VBM}(x)) / k_B T)$$
6. Solve Possison equation self-consistently (5 and 6)
   $$d^2E_{CBM}(x)/dx^2 = e(-\rho_e(x)+\rho_h(x)+N_D^+(x)-N_A^-(x)) /\varepsilon$$

How to calculate Fermi level E_F



$$E_g = E_C - E_V$$

Charge neutrality condition

$$N_A^- + N_e = N_D^+ + N_h \quad \Rightarrow \quad E_F$$

$$N_e = \int_{E_C}^{\infty} D_C(E) f_e(E, E_F) dE$$

$$N_D^+ = N_D\left[1 - f_e(E_D, E_F)\right]$$

フェルミ準位: 価電子帯、アクセプターを無視できる場合

$$f_e(E,E_F)=\frac{1}{1+\exp\left[\left(E-E_F\right)/k_BT\right]}$$

$$D_e(E)=\frac{\sqrt{2}}{\pi^2}\frac{m_{de}^{3/2}}{\hbar^3}\sqrt{E-E_C}$$

$E_C-E_F, E_D-E_F, >> k_BT$

$$N_e=\int_{E_C}^{\infty}D(E)f_e(E)dE\sim N_C\exp\left(-(E_C-E_F)/k_BT\right)$$

$$N_D^{+}=N_D\left[1-f_e(E_D,E_F)\right]\sim N_D\exp((E_D-E_F)/k_BT)$$

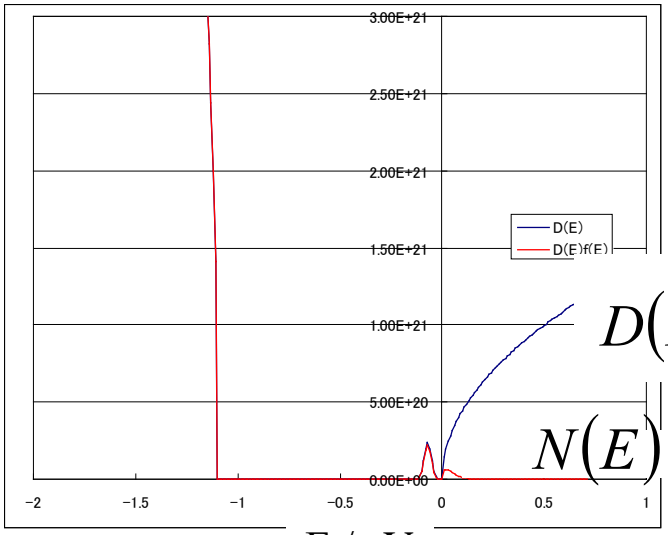
$$N_e=N_D^{+}$$

$$\exp(2E_F/k_BT)=N_D/N_C\exp((E_C+E_D)/k_BT)$$

$$E_F=\frac{E_C+E_D}{2}+\frac{k_BT}{2}\log(N_D/N_C)$$

How to calculate Fermi level E_F

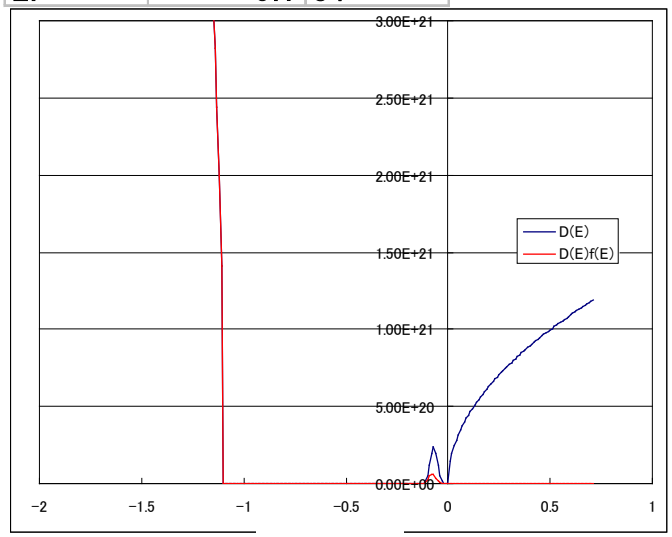
T=	300	K
EF=	0	eV
Nc=	5.20E+18	cm-3
Dc=	1.41E+21	
Ec=	0	eV
Nv=	5196000000	cm-3
Dv=	1.41E+22	
Ev=	-1.1	eV
ED=	-7.00E-02	eV
ND=	1.00E+19	cm-3
WD=	2.00E-02	eV
a2=	1.73E+03	
AD=	2.35E+20	



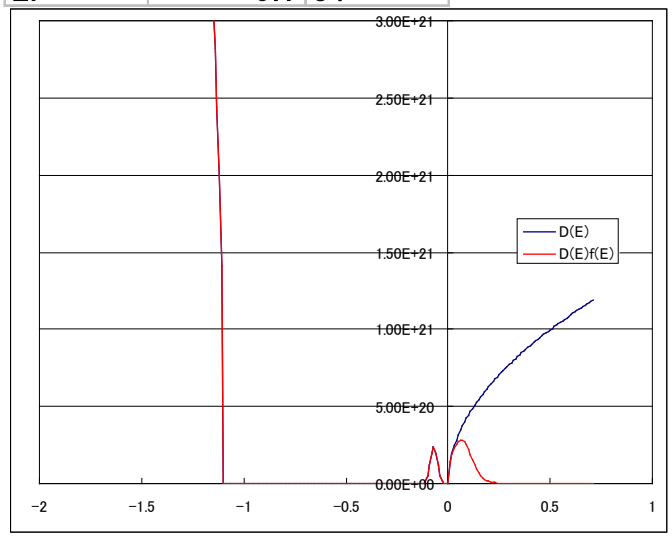
$$D(E)$$

$$N(E) = D(E)f(E)$$

EF=	-0.1	eV



EF=	0.1	eV



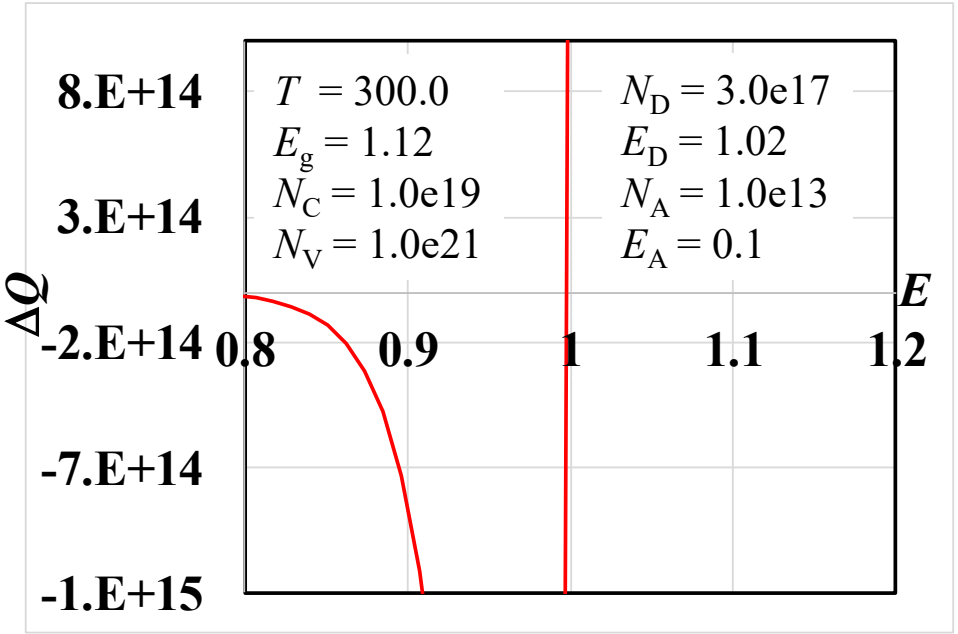
How to calculate E_F: Illustrative solution

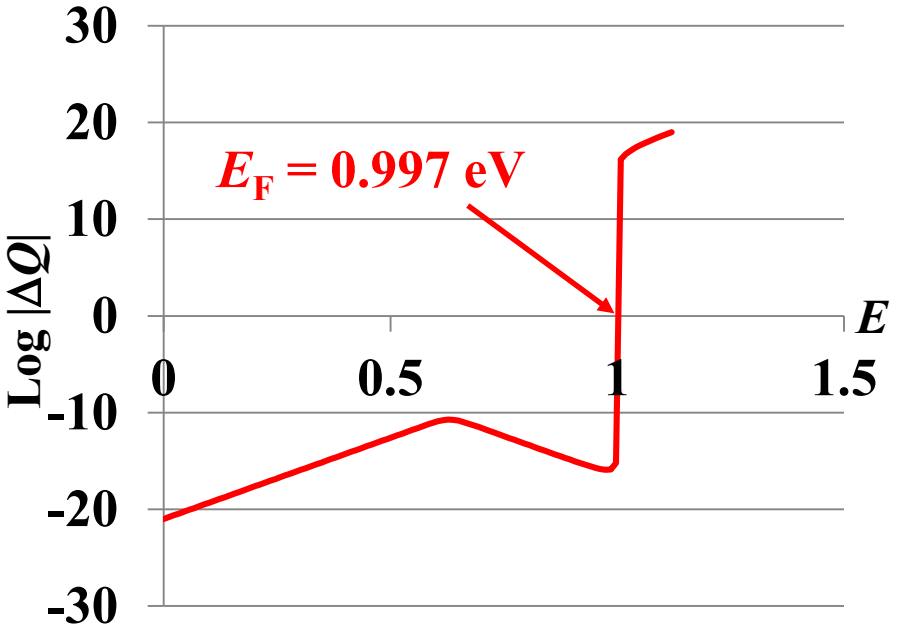
$$N_e = \int_{E_C}^{\infty} D_C(E) f_e(E, E_F) dE \qquad N_h = \int_{E_C}^{\infty} D_V(E) f_h(E, E_F) dE$$

$$N_D^+ = N_D\left[1 - f_e(E_D, E_F)\right] \qquad N_A^- = N_A\left[1 - f_h(E_A, E_F)\right]$$

$$f_h(E, E_F) = 1 - f_e(E, E_F)$$

Plot $\Delta Q = (N_A^- + N_e) - (N_D^+ + N_h)$ w.r.t. E_F and find $\Delta Q = 0$





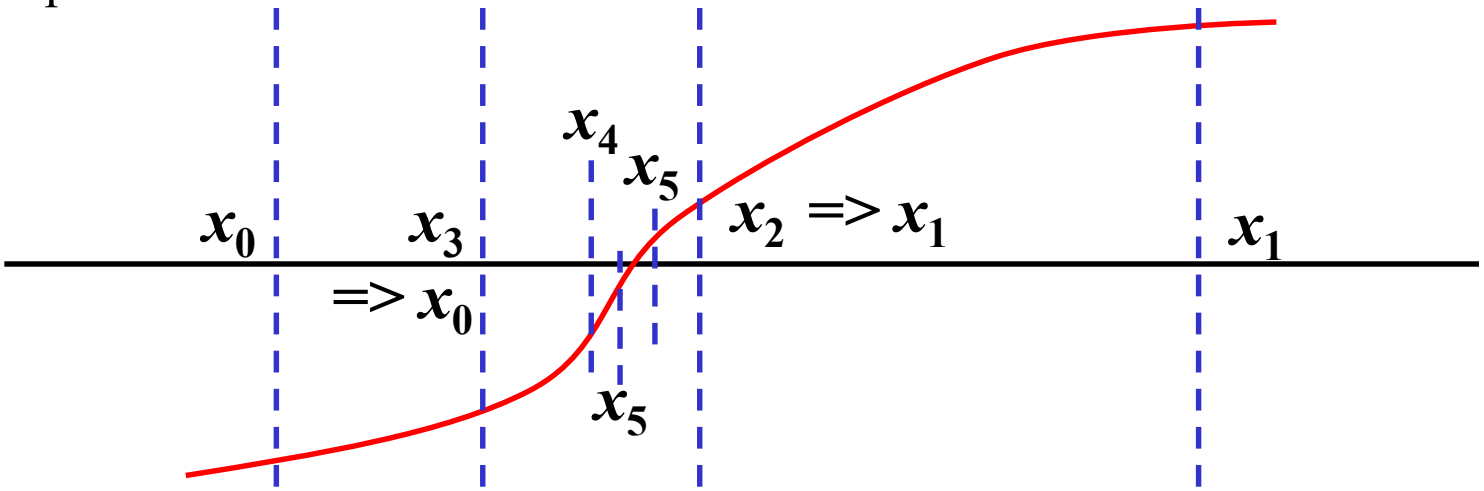
Bisection method (二分法): Monotonic func (単調関数)

Solution of $f(x) = 0$ for monotonic function $f(x)$

1. **Start from a range $[x_0, x_1]$ where $f(x_0) < 0$ & $f(x_1) > 0$**
   (or $f(x_0) > 0$ & $f(x_1) < 0$)
   *** Solution exist in this range for a monotonic function**
2. **Solve the equation by the following iterative procedure**

Case $f(x_0) < 0$ and $f(x_1) > 0$: Judge by $f(x_0) \cdot f(x_1) < 0$

1. $x_2 = (x_0 + x_1) / 2.0$
2. If $f(x_2) > 0$ ($f(x_0) \cdot f(x_2) < 0$), x_1 is replaced with x_2
   If $f(x_2) < 0$ ($f(x_1) \cdot f(x_2) < 0$), x_0 is replaced with x_2
3. Solution x_2 is obtained when $|x_1 - x_0|$, $|f(x_1) - f(x_0)|$ becomes less than EPS.
4. Repet 1 – 3



Fermi level in semi.: python program

Program: EF-T-semiconductor.py

http://conf.msl.titech.ac.jp/Lecture/StatisticsC/EF-T-semiconductor.html

Usage: python EF-T-semiconductor.py EA NA ED ND Ec Nv Nc

Run: python EF-T-semiconductor.py 0.05 1.0e15 0.95 1.0e16 1.0 1.2e19 2.1e18

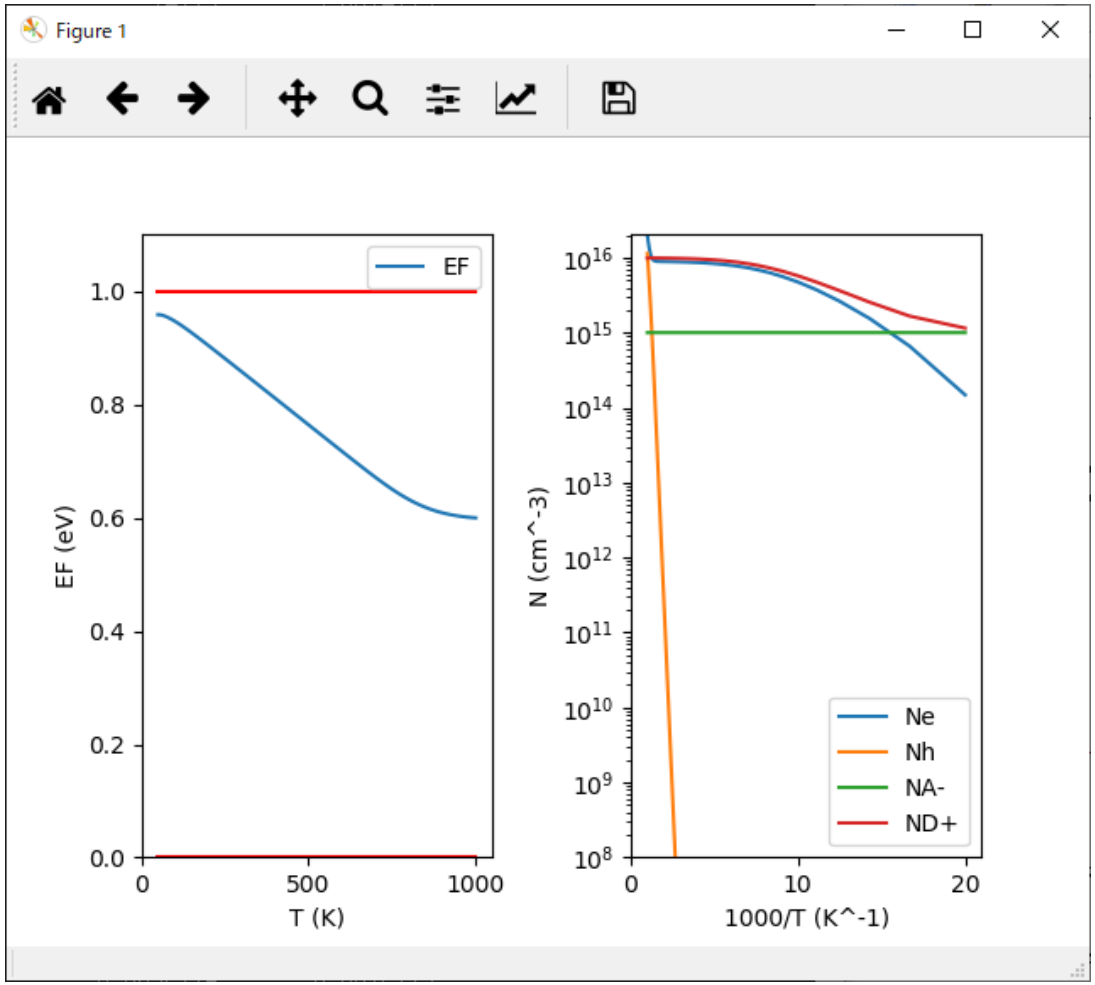
$E_c = 0$, $E_c = 1.0$ eV (= band gap)

$E_A = 0.05$ eV, $N_A = 10^{15}$ cm^{-3},

$E_D = 0.95$ eV, $N_D = 10^{16}$ cm^{-3}

$N_c = 1.2 \times 10^{19}$ cm^{-3}

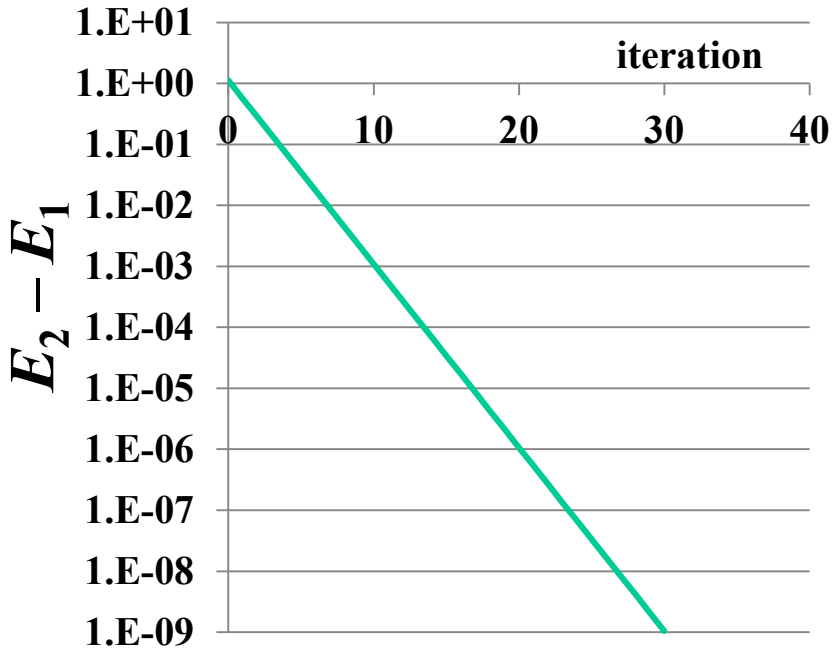
$N_v = 2.1 \times 10^{18}$ cm^{-3}

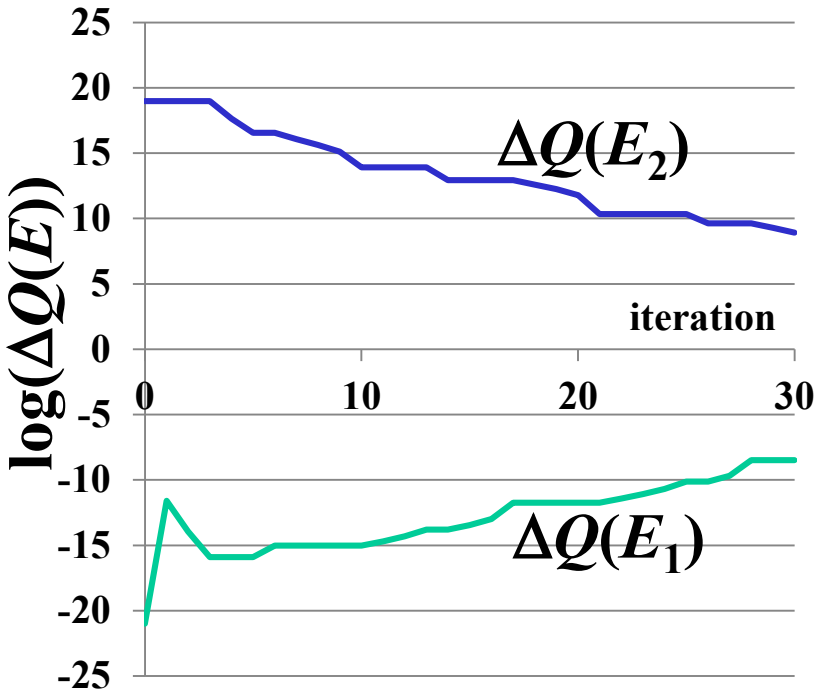


E_F by bisection method: Convergence procedure

Initial range: $[E_1, E_2] = [E_V = 0, E_C = E_g]$

Find $\Delta Q = (N_A^- + N_e) - (N_D^+ + N_h) = 0$





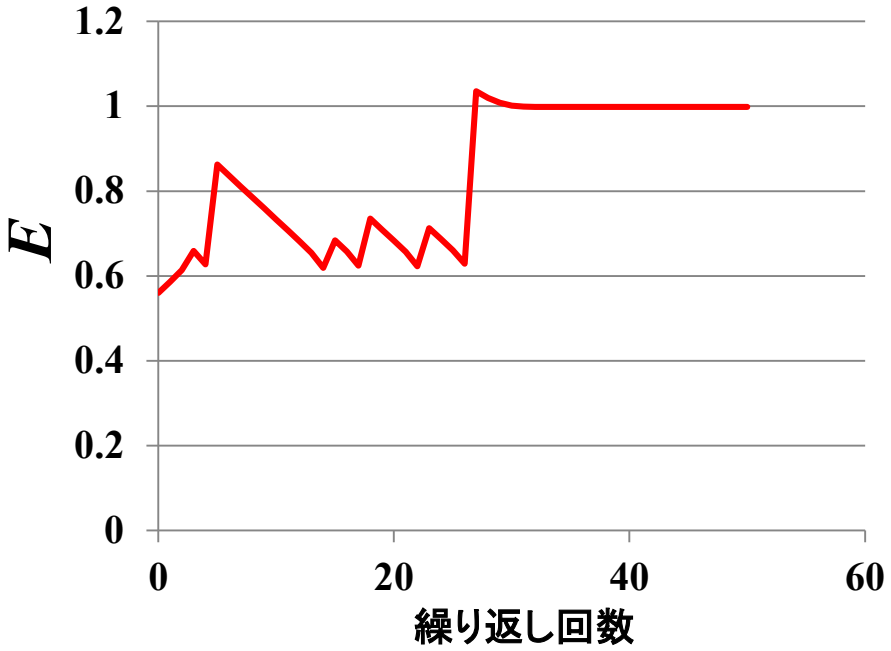
After 30 times iterations

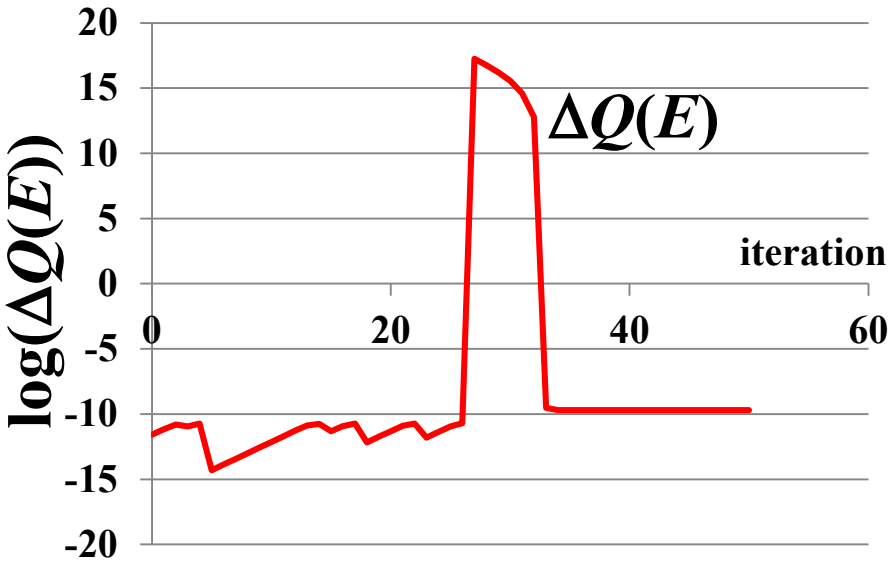
$E_F = [0.9985173589, 0.9985173599]$

$dQ = [-3\times10^8, 8\times10^8]$

How to calculate E_F: Newton-Raphson

Use initial value $E_0 = (E_V + E_C) / 2.0$
and find $\Delta Q = (N_A^- + N_e) - (N_D^+ + N_h) = 0$





After 30 times iterations

E_F = 0.998517354556472

$dQ = -5\times10^9$

Solution of multi-parameter simultaneous equations

Solve $f_l(x_k) = 0$

$$f_l(x_k + \delta x_k) \sim f_l(x_k) + \sum_{k'} \delta x_{k'} \frac{\partial f_l(x_k)}{\partial x_{k'}} = 0$$

$$\begin{pmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,N} \\ f_{2,1} & f_{2,2} & & f_{2,N} \\ \vdots & & \ddots & \vdots \\ f_{n,1} & f_{n,N} & \cdots & f_{n,N} \end{pmatrix} \begin{pmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_N \end{pmatrix} = - \begin{pmatrix} f_1(x_k) \\ f_1(x_k) \\ \vdots \\ f_n(x_k) \end{pmatrix}$$

$$f_{l,k'}(x_k) = \frac{\partial f_l(x_k)}{\partial x_{k'}}$$

Unique solution may be obtained when $n = N$

一意的に解ける可能性があるのは $n = N$ の時

For many practical problems:

1. Data are much larger than the number of parameters ($N < n$)

2. Target: Find x_k to satisfy small $f_l(x_k)$

=> *e.g.* LSQ is more appropriate

Non-linear (NL) optimization

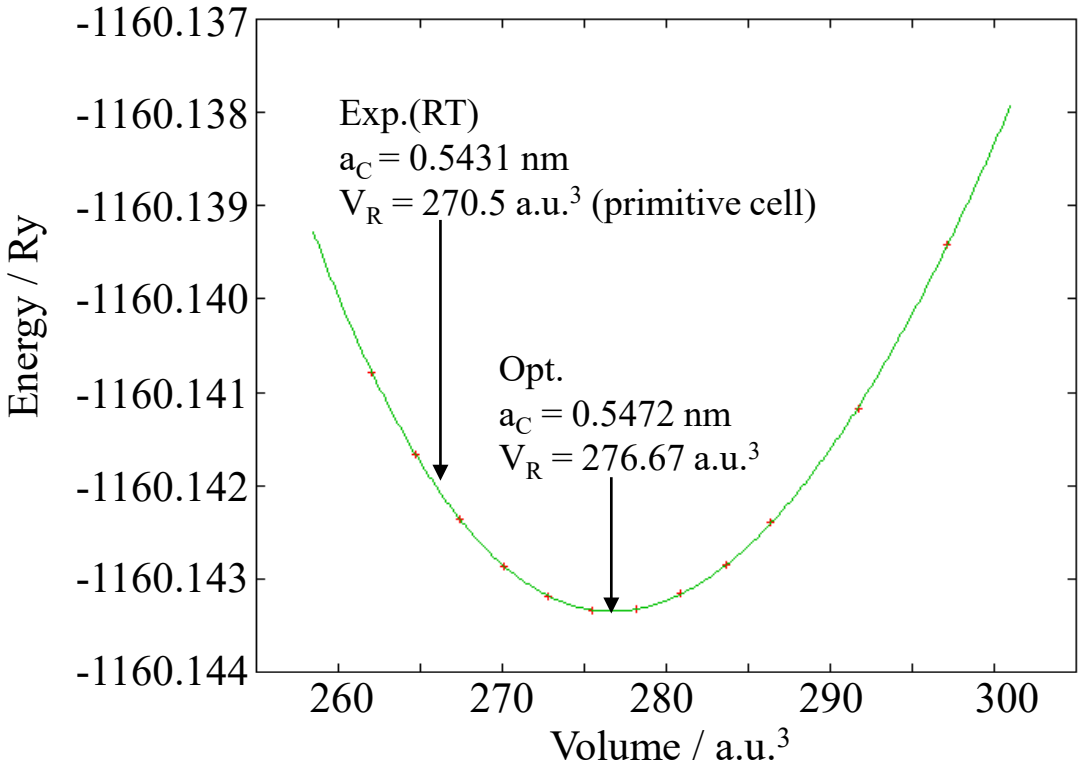
非線形最適化

NL optimization of crystal structure: Illustrative approach

安定構造: 図解による解法

Calculate total energy by quantum calculations by varying a lattice parameter

***ex.* Si**



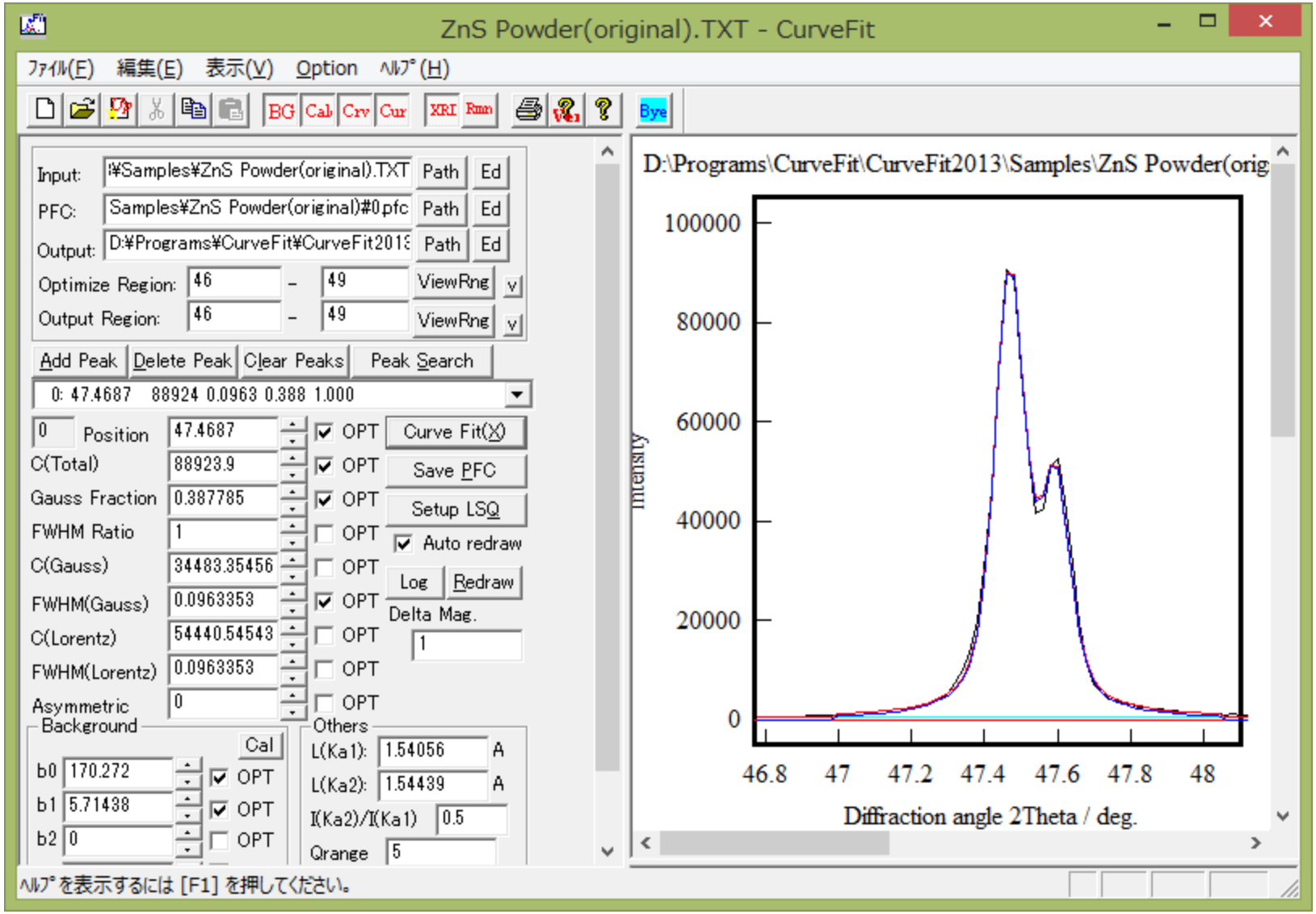
$$E = E_{\min} + 1/2 B_0 \left(V / V_0\right)^2$$

B_0 (GPa) = 87.57 GPa (exp: 97.88 GPa)

Ex.: Deconvolution of powder XRD peak

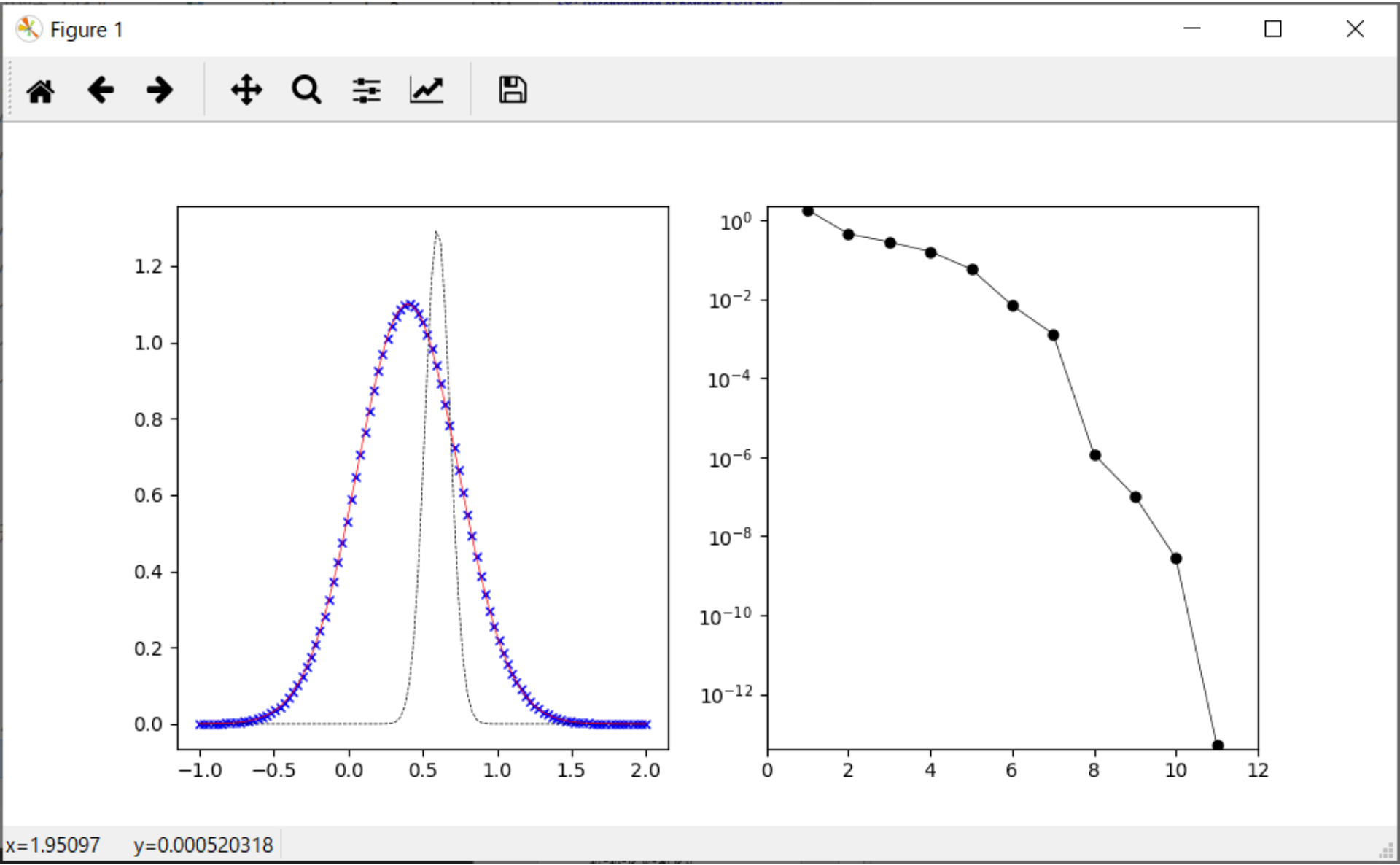
peakfit.exe

Incorporate the intensity ratio from $K\alpha_1$ and $K\alpha_2$ at 2:1



Ex.: Deconvolution of powder XRD peak

peakfit-scipy-minimize.py



Profile models used for spectroscopy

Lorentz function

$$I_L(x) = \frac{1}{1 + \left[(x - x_0)/w\right]^2}$$

***w*: half width at half maximum**

Gauss function

$$I_G(x) = \frac{1}{a_w w \pi^{1/2}} \exp\left\{-\left[(x - x_0)/(a_w w)\right]^2\right\}$$

$$a_w = (\ln 2)^{-1/2} = 0.832554611$$

Voigt function:

E.g., observed is convolution of sample spectrum $I_L(x)$ and apparatus function $I_G(x)$

$$I_V(x) = \int_{-\infty}^{\infty} I_G(x') I_L(x - x') dx'$$

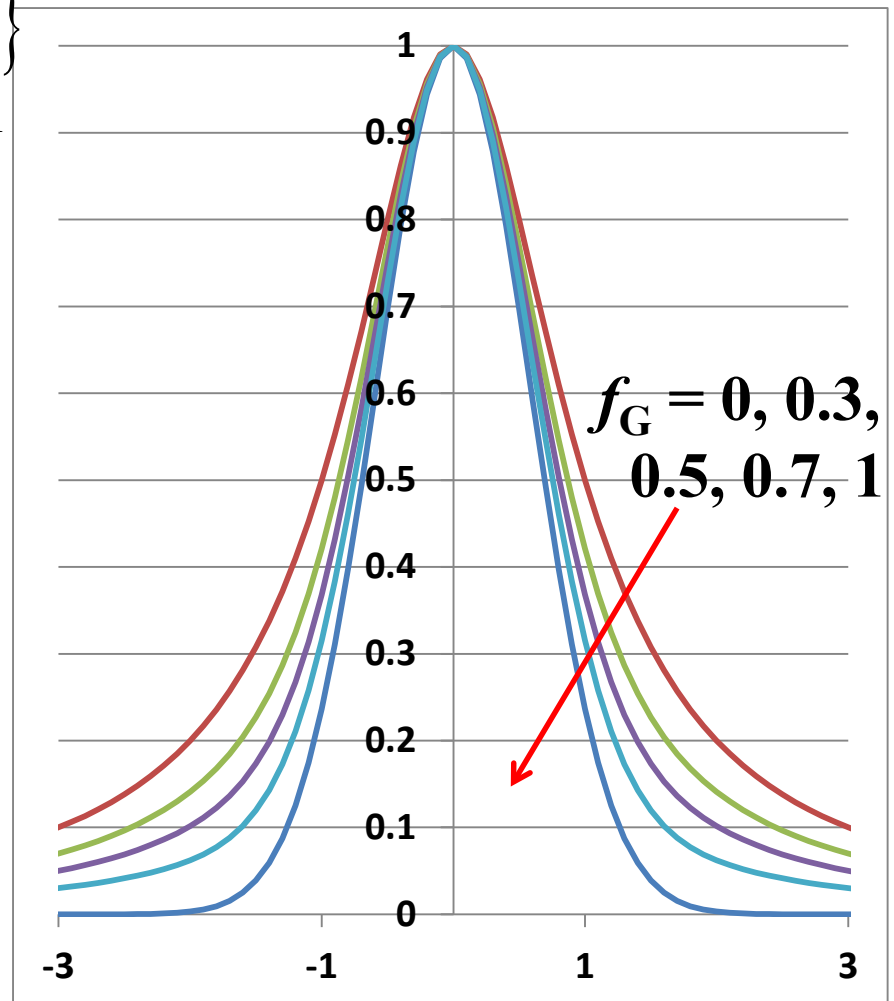
$$= \frac{a_V}{\pi} \int_{-\infty}^{\infty} \frac{\exp(-x'^2)}{a_V{}^2 + (x - x')^2} dx'$$

Pseudo-Voigt function:

Simplified Voigt function

$$I_{PV}(x) = f_G I_G(x) + (1 - f_G) I_L(x)$$

f_G: Gauss fraction



Multiple parameter Newton-Raphson method

Extend to multiple parameters: Minimize $F(x_l)$

$$f_k(x_l) = \partial F(x_l)/\partial x_k = 0$$

Iteration: $f_k(x_l + \delta x_l) \sim f_k(x_l) + \sum_{k'} \delta x_{k'} \partial f_k(x_l)/\partial x_{k'} = 0$

$$x_{l,1} = x_{l,0} - (\partial f_k(x_l)/\partial x_{k'})^{-1}(f_k) = x_{l,0} - (F''_{kk'})^{-1}(F'_k)$$

$$F''_{kk'} = \frac{\partial^2 F(\mathbf{x})}{\partial x_k \partial x_{k'}}$$

Hessian matrix (ヘッセ行列)
(ヘッセ行列の固有値をヘッシアンと呼ぶ)

Hessian matrix is not always positive definite (正定値であるとは限らない)
(Maximum, Saddle point 極大値、鞍点**)**
=> *F*" dose not always gives decreasing direction

Convert *F*" to positive definite and suppress divergence

$$x_{l,1} = x_{l,0} - (F''_{kk'} + \lambda I)^{-1}(F'_k)$$

λ: Dumping Factor

Steepest Descend (SD) method (最急降下法)

矢部博, 工学基礎 最適化とその応用, 数理工学社 (2006)

Search minimum only by first derivatives. Simplest one among differential methods

- **SD: S^2 would decrease in the vector $-(df / dx_i)dx_i$**

  $x_i^{(i+1)} = x_i^{(i)} - \alpha(df / dx_i)$

  α_k may be a small constant step

  or determined by a line search method

***ex.* in right figure:**

$S^2 = f(x_i) = 5x_1^2 + x_2^2$, initial $x_1 = 0.7, x_2 = 1.5$

- **Newton method**

  One cycle calculation provides the final solution for quadratic problems

  楕円問題の場合は一度目の計算で最適値に到達

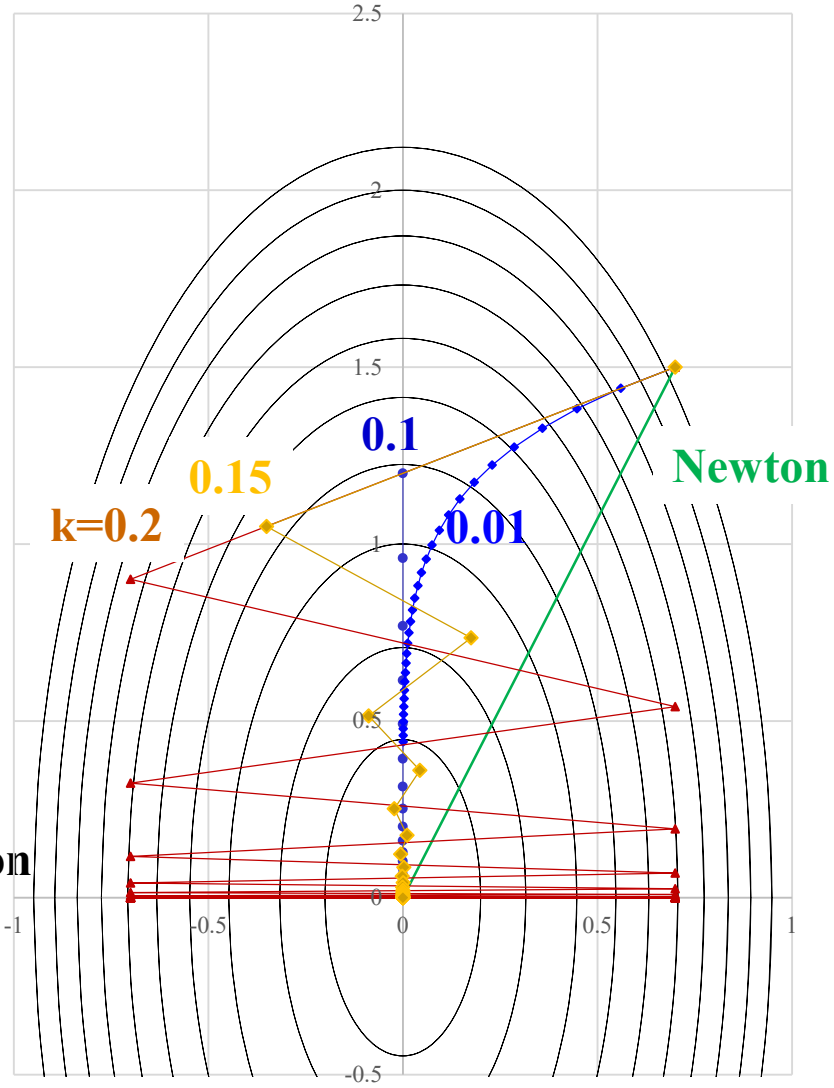
- **SD method**

  α = 0.3: Diverged (not shown in the graph)

  0.2, 0.15: Converged, but oscillated

  0.1: Reach final solution by one cycle calculation

  0.01: Not oscillated, but slowly converged



Problem: If S^2 is highly anisotropic, the SD direction would be different largely from the minimization direction S^2 が大きく非対称な場合、最急勾配方向は最小値方向とは大きく異なることがある

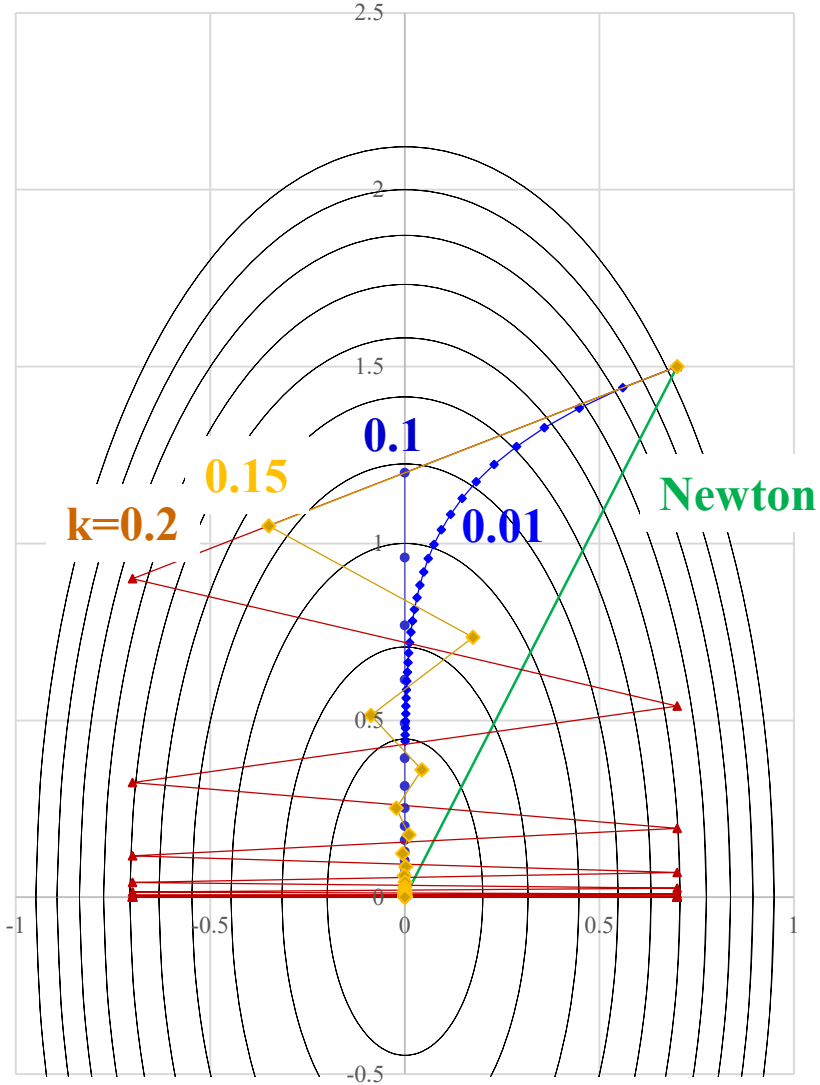
=> **Conjugate Gradient (CG)** method (共役勾配法)

Steepest Descend method

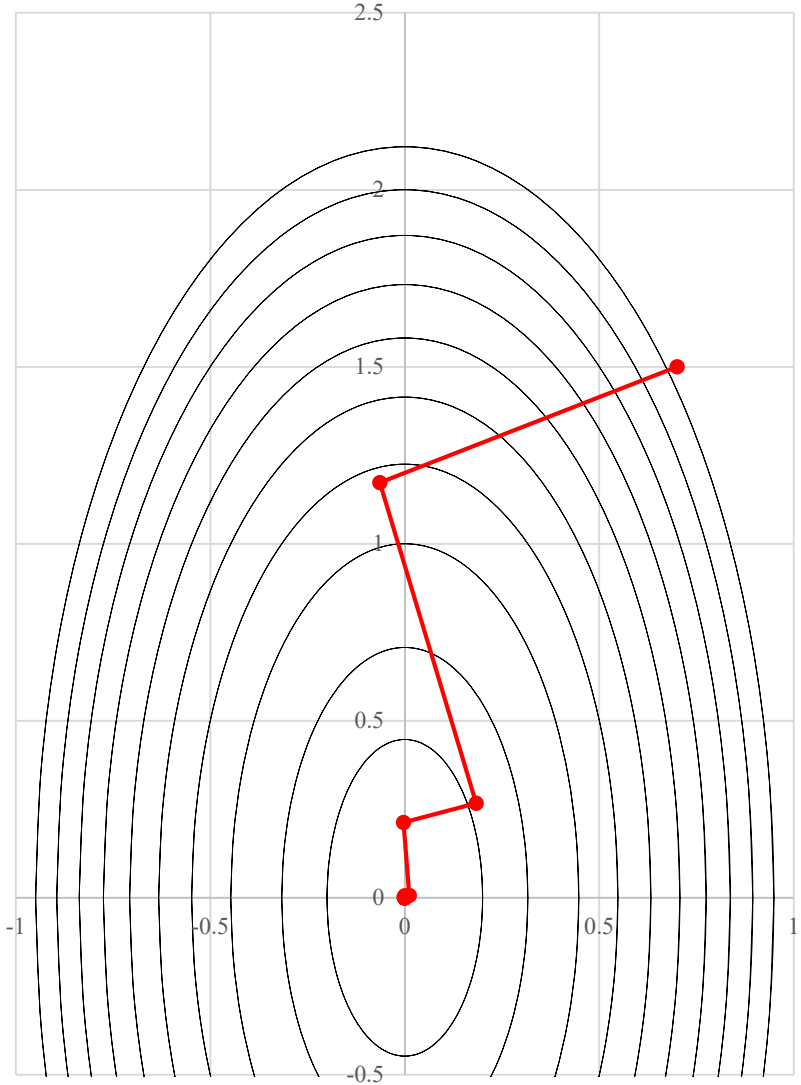
矢部博, 工学基礎 最適化とその応用, 数理工学社 (2006)

Effective way: α is determined by line search (直線探索法)

Without direct search



By direct search



SD method in Deep Learning

- **All batch data are divided to mini batches, and apply SD to each mini batch**

Right example:

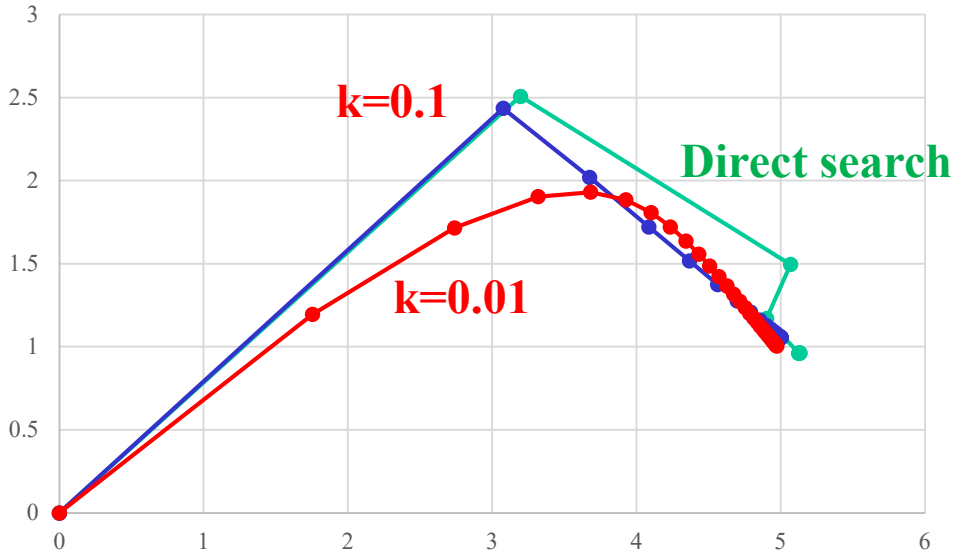
$S^2 = f(x_i) = ax_1^2 + bx_2^2, \quad a = 5, b = 1$

1000 batch data are generated with random num

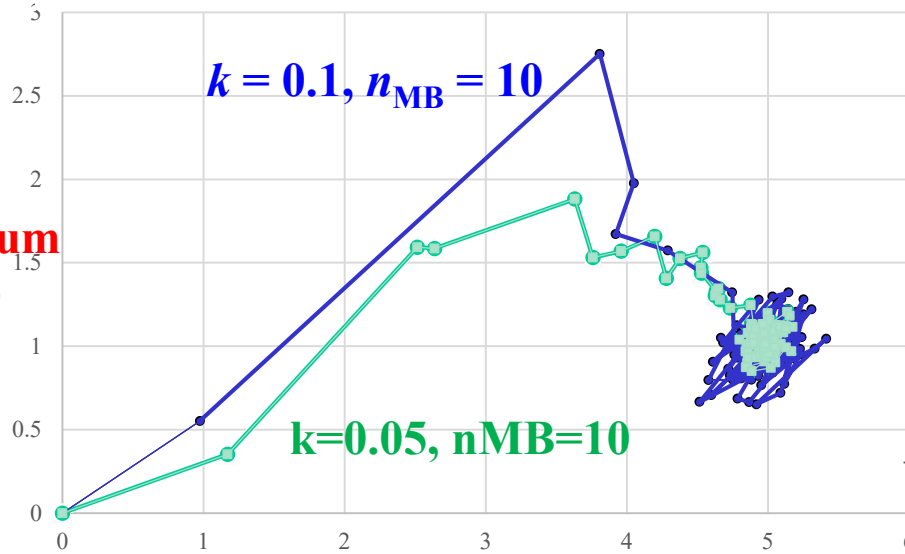
(note: the data were re-generated for different runs)

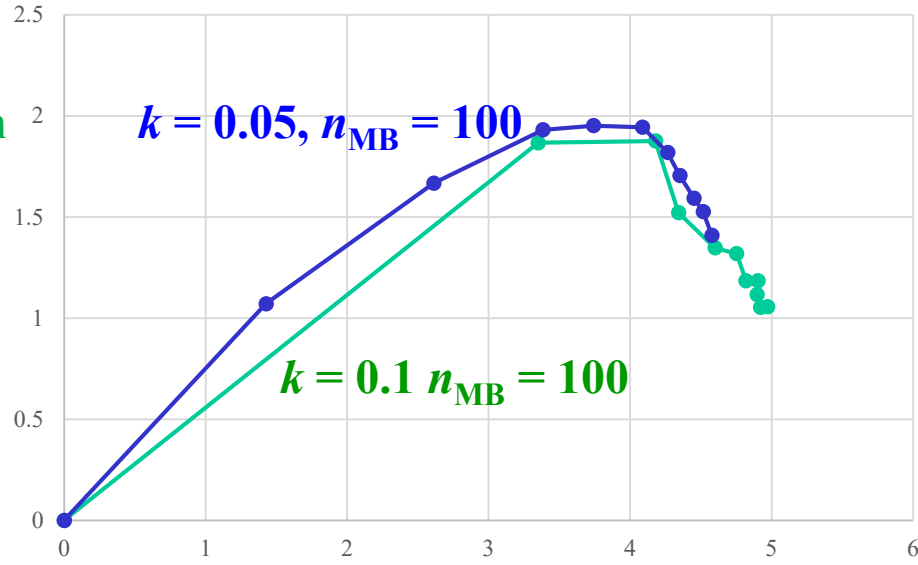
Initial $a = 0, b = 0$

SD (Repeat for all batch data)



DL: SD method





Conjugate Gradient method (共役勾配法)

矢部博, 工学基礎 最適化とその応用, 数理工学社 (2006)

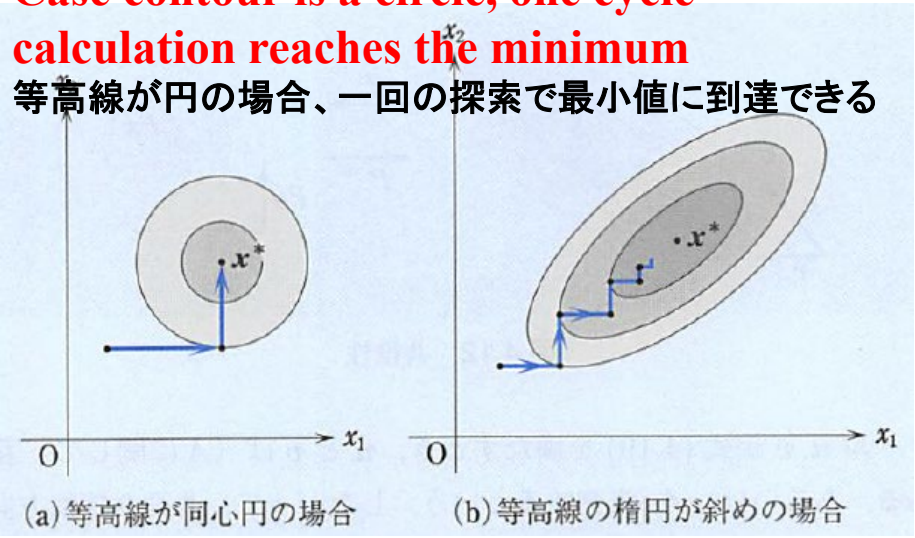
Vectors u and v satisfy $u^t Av = 0$ *for a matrix A: u* and *v and conjugate with each other*

- **For quadratic function, repetition of the conjugate direction will find the minimum in finite cycles if exact line search is employed**

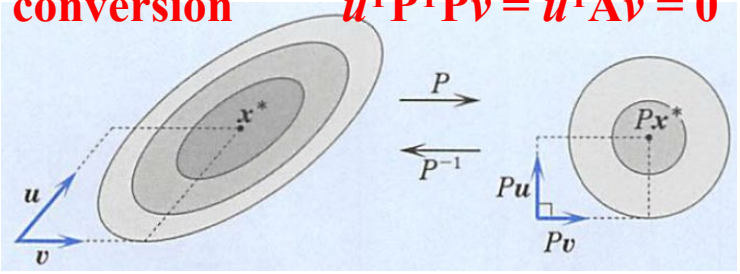
  共役な探索方向に沿って正確な直線探索を実行 => 有限回の反復で2次関数の最小解に到達

Case contour is a circle, one cycle calculation reaches the minimum

等高線が円の場合、一回の探索で最小値に到達できる



Conjugate vectors and ellipsoido – circle conversion $u^{\mathrm{T}}\mathrm{P}^{\mathrm{T}}\mathrm{P}v = u^{\mathrm{T}}\mathrm{A}v = 0$



1. **Give initial value x_0**
2. **Initial direction d is determined by SD**

$$\mathbf{d} = -\nabla f$$

3. **Find x_{k+1} using appropriately chosen α_k**

$$x_{k+1} = x_k + \alpha_k d_k$$

α_k may be a small constant step or determined by a line search method

4. **Search direction is updated by**

$$\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$$

$$\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \frac{\nabla f(\mathbf{x}_{k+1})^T \mathbf{y}_k}{\mathbf{d}_k^T \mathbf{y}_k}\mathbf{d}_k$$

5. **Repeat 3 – 4 to reach convergence**

As the freedom of cg directions is the number of parameters (n_{param}), need to go back to **2** to reset $\boldsymbol{d}_k$ at some interval (typically n_{param}, necessary for $n_{\mathrm{param}} = 2$).

Marquart method (マーカート法)

Minimize a square sum of *m* functions $f_j(x_i)$ with *N* parameters

$$F(x_i) = \sum_{j=1}^{m} f_j(x_i)^2$$

Approximate by

$$f_j(x_i + \delta x_i) \sim f_j(x_i) + \left(\frac{\partial f_j}{\partial x_k}\right)(\delta x_i) = f_j(x_i) + \mathbf{A}\boldsymbol{\delta} x_i \qquad A_{jk} = \frac{\partial f_j}{\partial x_k}$$

$$F(x_i + \delta x_i) \sim F(x_i)^2 + 2\sum_{j,k} f_j A_{jk} \delta x_k + \sum_{j,k,k'} A_{jk} A_{ik'} \delta x_k \delta x_{k'}$$

$$\frac{\partial F(x_i)}{\partial \delta x_k} \sim 2\sum_j \left(A_{jk} f_j + \sum_k A_{ik} A_{jk} \delta x_j \right) = 0$$

$$\boldsymbol{\delta} x = -\left(\mathbf{A}^t \mathbf{A}\right)^{-1} \mathbf{A}^t \left(f_j\right)$$ **Gauss-Newton method**

Levenberg-Marquart method

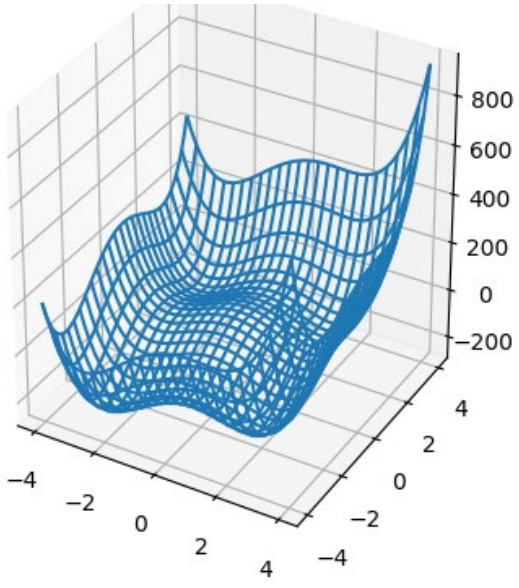
$$\boldsymbol{\delta} x = -\left(\mathbf{A}^t \mathbf{A} + \lambda I\right)^{-1} \mathbf{A}^t \left(f_j\right)$$ **λ: dumping factor**

$$\boldsymbol{\delta} x = -\left(\mathbf{A}^t \mathbf{A} + \lambda \mathrm{diag}(\mathbf{A}^t \mathbf{A})\right)^{-1} \mathbf{A}^t \left(f_j\right)$$

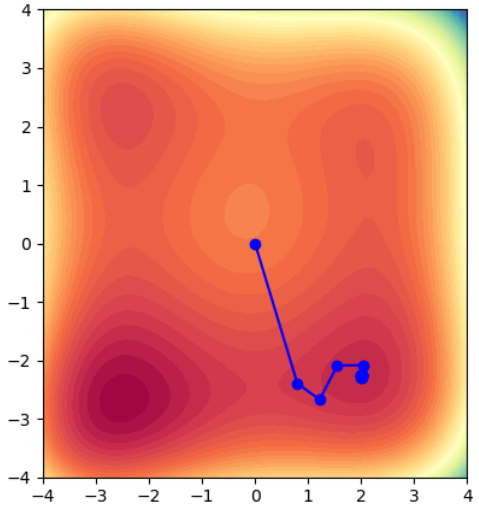
e.g. chosen proportional to diagonal sum of $\mathbf{A}^t\mathbf{A}$

Comparison

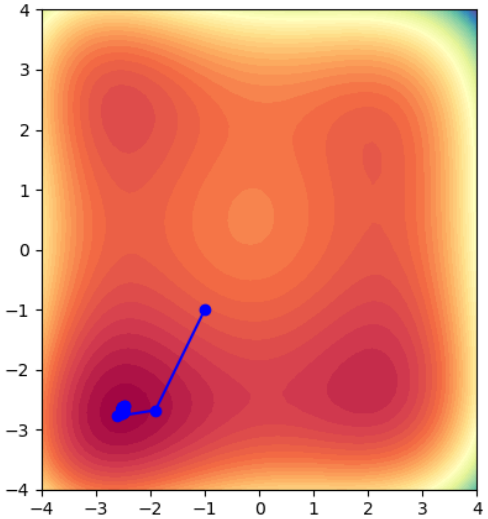
http://conf.msl.titech.ac.jp/Lecture/python/index-numericalanalysis.html
optimize-sd-cg2d-linesearch.py, optimize-newton-raphson2d.py



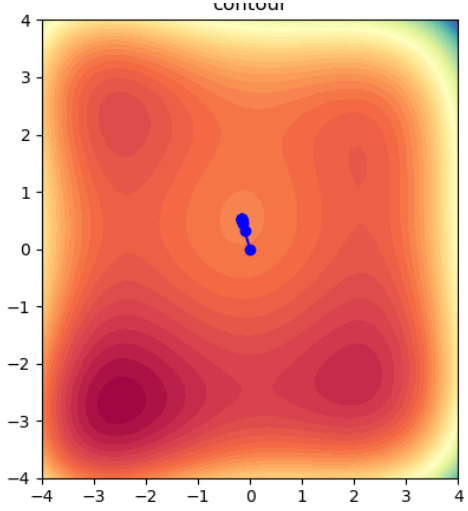
From (0.0 0.0) cg simple



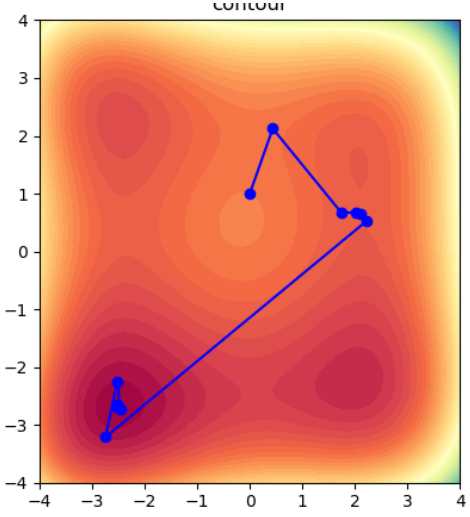
From (-1.0 -1.0) cg simple



From (0.0 0.0) Newton



From (0.0 1.0) SD armijo



Features of NL optimization

- **Newton-Raphson method:**
  Use second derivatives (Hessian matrix)
  Fast convergence, easily diverged, complex program
- **Steepest Descent:**
  Use first derivatives only
  Simple program, Slower convergence than NR and CG
- **Conjugate Gradient:**
  Use conjugate direction for efficient search
  Better convergence than NR, faster than SD, complex program
- **Marquart:**
  Use first derivatives of $f_j(x_i)$
  Simple program, Slower convergence than NR
- **Simplex:**
  Trial and error with a pre-determined selections of next candidate parameters
  Very slow but good convergence

Simplex method (単体法, Amoeba法)

服部力、名取亮、小国力 監修、Fortranによる数値計算ソフトウェア、丸善株式会社（1989年）

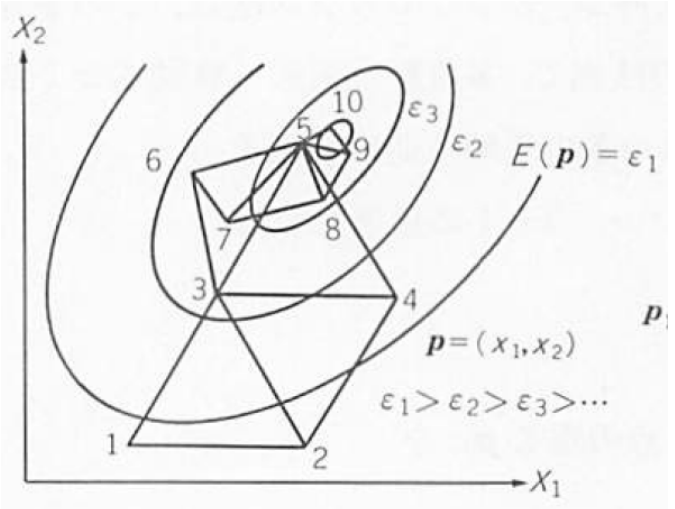
Simplex: Polyhedron formed by (n+1) vertexes in n-dimension space
(単体: n次元空間で(n+1) 個の頂点が作る多面体)

Minimize $F(\boldsymbol{x}_i)$

1. (n+1) initial values $\boldsymbol{x}_i$ (i = 1, 2, ···, n+1) => Sort $F(\boldsymbol{x}_i)$ so that $F(\boldsymbol{x}_i) > F(\boldsymbol{x}_{i'})$ ($i < i'$)
   $\boldsymbol{x}_h = \boldsymbol{x}_1$, $\boldsymbol{x}_l = \boldsymbol{x}_{n+1}$
2. Average except the maximum vertex $\boldsymbol{x}_i$ $\quad x_G = \sum_{i=2} x_i / n$
3. New x will be examined along the line $\boldsymbol{x}_1 - \boldsymbol{x}_G$ by the following selections

(i) Reflection (鏡映) : $\boldsymbol{x}_R = (1+\alpha)\boldsymbol{x}_G - \alpha\boldsymbol{x}_1$ ($\alpha > 0$, *ex.* 1.0)
(ii) Expansion (拡大) : $\boldsymbol{x}_E = \gamma\boldsymbol{x}_r + (1-\gamma)\boldsymbol{x}_G$ ($\gamma > 0$, *ex.* 2.0)
(iii) Contraction (収縮) : $\boldsymbol{x}_C = \beta\boldsymbol{x}_1 + (1-\beta)\boldsymbol{x}_G$ ($0 < \beta < 1$, *ex.* 0.5)
(iv) Reduction (縮小) : $\boldsymbol{x}_{RD} = (\boldsymbol{x}_1 + \boldsymbol{x}_l) / 2$

4. Replace $\boldsymbol{x}_1$ with the x in (i) – (iv) that firstly satisfies $F(\boldsymbol{x}) < F(\boldsymbol{x}_1)$
5. Repeat 2 - 4



Simplex method (単体法, Amoeba法)

南茂夫 編著、科学計測のための波形データ処理、CQ出版 (1986年)

Simplex: Polyhedron formed by (n+1) vertexes in n-dimension space
(単体: n次元空間で (n+1) 個の頂点が作る多面体)

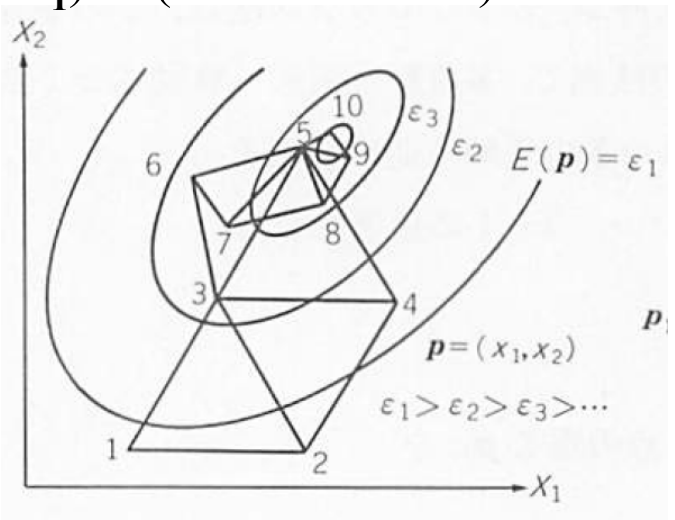
Minimize $F(\boldsymbol{x}_i)$

1. (n+1) initial values $\boldsymbol{x}_i$ (i = 1, 2, ···, n+1) => Sort $F(\boldsymbol{x}_i)$ so that $F(\boldsymbol{x}_i) > F(\boldsymbol{x}_{i'})$ ($i < i'$)
2. Average except the maximum vertex $\boldsymbol{x}_i$ $x_G = \sum_{i=2} x_i / n$

3. New x will be examined along the line $\boldsymbol{x}_1 - \boldsymbol{x}_G$ by the following selections

(i) Reflection (鏡映) : $\boldsymbol{x}_R = \boldsymbol{x}_1 + \alpha(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (*ex.* $\alpha = 2$)
(ii) Expansion (拡大) : $\boldsymbol{x}_E = \boldsymbol{x}_1 + \beta(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (*ex.* $\beta > 2$)
(iii) Reduction-Reflection (縮小鏡映) : $\boldsymbol{x}_{CR} = \boldsymbol{x}_1 + \gamma(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (*ex.* $1.0 < \gamma < 2.0$)
(iv) Reduction : $\boldsymbol{x}_{CW} = \boldsymbol{x}_1 + \delta(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (*ex.* $0 < \delta < 1.0$)

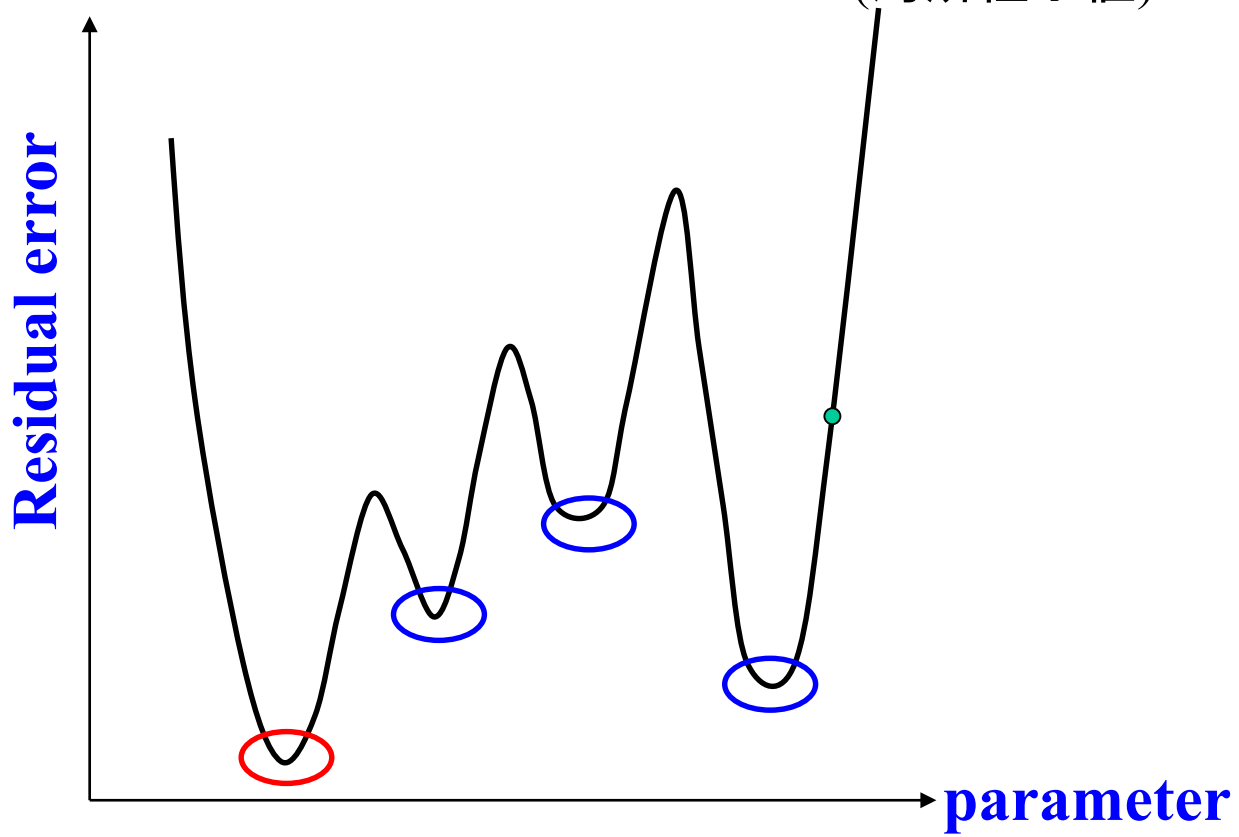
4. Replace $\boldsymbol{x}_1$ with the x in (i) – (iv) that firstly satisfies $F(\boldsymbol{x}) < F(\boldsymbol{x}_1)$
5. Repeat 2 - 4



Notes for NL optimization: Local minimum

- Solutions may be more than one
- Final solution is not obtained by one step calculation
- **Convergence must be confirmed**
- **Confirm the solution is the global minimum** (大域最小値)

  ⇔ **Often fall in a local minimum** (局所極小値)



Notes for NL optimization: Over-fitting

MR-TwoCarriermodel.py (to be uploaded for Lab only web)

e.g. in APL 107, 182411 (2015)

$$\rho_{xx}(B) = \mathrm{Re}(\boldsymbol{\rho}) = \frac{1}{e}\frac{(n_h\mu_h + n_e\mu_e) + (n_h\mu_e + n_e\mu_h)\mu_h\mu_e B^2}{(n_h\mu_h + n_e\mu_e)^2 + (n_h - n_e)^2\mu_h^2\mu_e^2 B^2},$$

$$\rho_{yx}(B) = -\mathrm{Im}(\boldsymbol{\rho}) = \frac{B}{e}\frac{(n_h\mu_h^2 - n_e\mu_e^2) + (n_h - n_e)\mu_h^2\mu_e^2 B^2}{(n_h\mu_h + n_e\mu_e)^2 + (n_h - n_e)^2\mu_h^2\mu_e^2 B^2}.$$

# of parameters must be larger than # of constraints,

# of parameters: four n_h, μ_h, n_e, μ_e

# of constraints: three

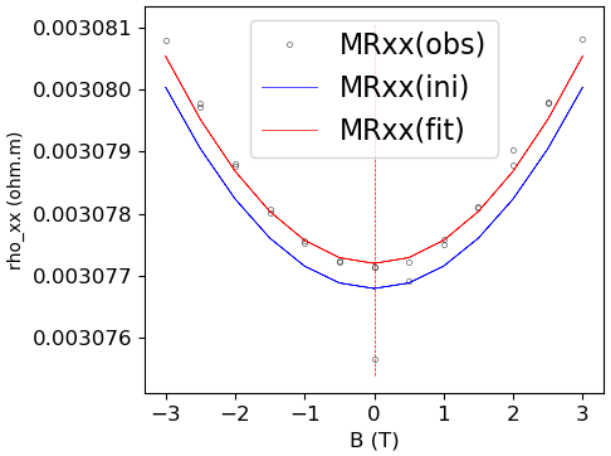
For parabolic $\rho_{xx}(B)$: $a_{xx}^0 = \rho_{xx}(0)$, and a_{xx}^2

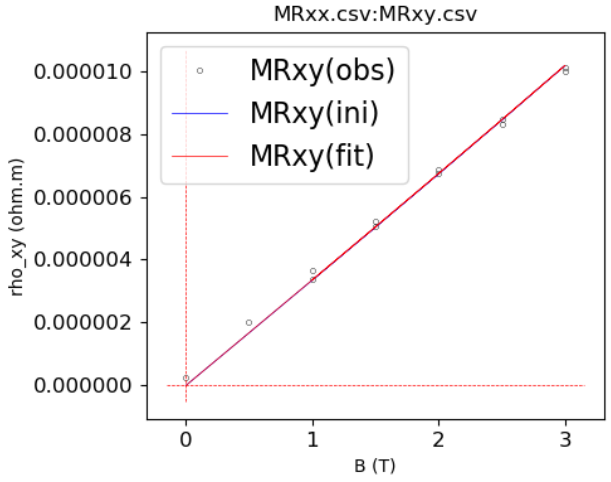
For linear $\rho_{xy}(B)$: a_{xy}^1 only

At least two parameter sets gives similar residuals S^2

$n_h = 2.09\text{x}10^{23}$ m^{-3}
$\mu_h = 0.074$ m^2/Vs
$n_e = 2.72\text{x}10^{22}$ m^{-3}
$\mu_e = 0.018$ m^2/Vs
$S^2 = 3.2\text{x}10^{-14}$

$n_h = 9.23\text{x}10^{22}$ m^{-3}
$\mu_h = \mu_e = 0.012$ m^2/Vs
$n_e = 7.68\text{x}10^{22}$ m^{-3}
$S^2 = 3.0\text{x}10^{-14}$





Features of NL optimization algorisms

Convergence	A	B
Speed	×	○
Stability	○	×
Region	○	×
For:	Initial cycles	Later fast convergence

A: Simplex (単体法)

A,B: Conjugate Gradient (CG, 共役勾配法)

B: Steepest Descent (SD, 最急降下法)

B: Newton-Raphson method, Quasi Newton methods

- **Davidson-Fletcher-Powell (DFP)**
- **Broyden-Fletcher-Goldfarb-Shanno (BFGS)**