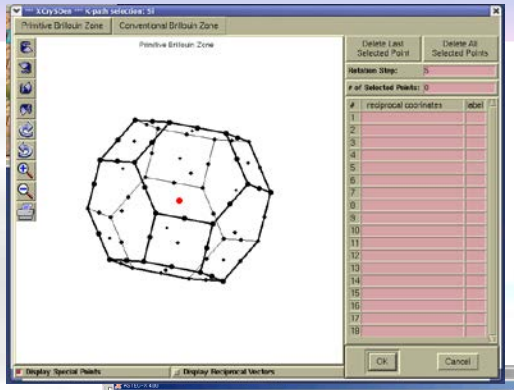
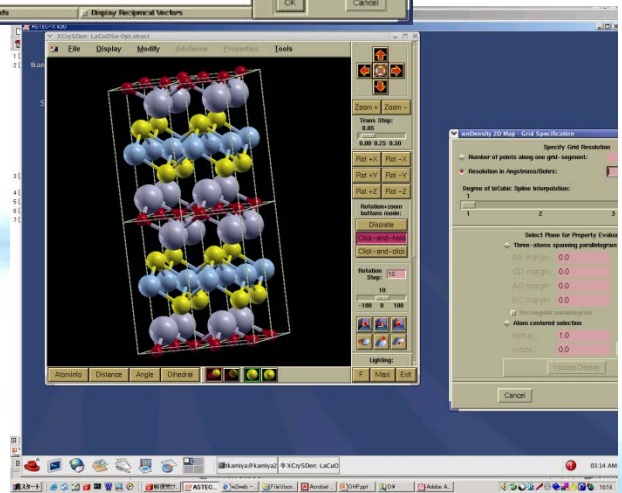
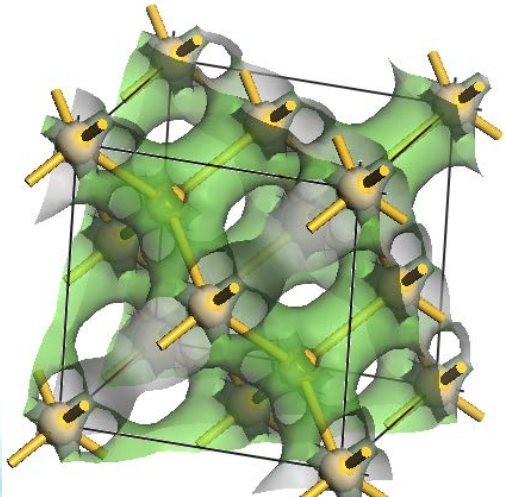
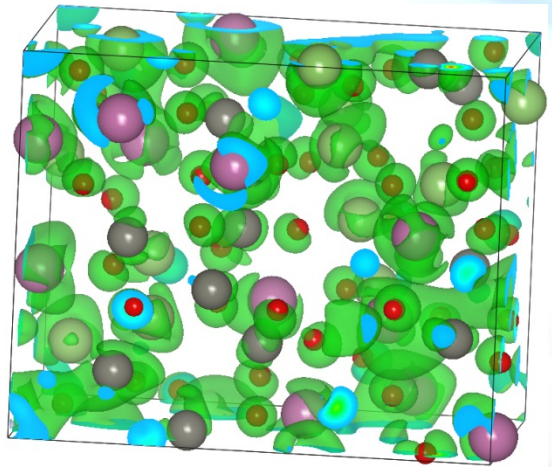
計算材料科学特論

神谷利夫









講義予定

神谷担当講義資料配布: http://conf.msl.titech.ac.jp/Lecture/

6月12日	神谷1 (コンピュータの原理)
6月15日	神谷2 (数値微分・積分)
6月19日	神谷3 (常微分方程式、分子動力学法、補間、平滑化)
6月22日	神谷4 (平滑化、線形最小自乗、最適化、方程式の数値解)
6月26日	神谷5 (非線形最小自乗、Fourier変換)
6月29日	笹川1 (量子論おさらい1)
7月 3日	若井1 (微構造形成への応用、Monte Carlo法その1)
7月 6日	笹川2 (量子論おさらい2)
7月10日	若井2 (微構造形成への応用、Monte Carlo法その2)
7月13日	笹川3 (第一原理計算:基礎)
7月17日	若井3 (微構造形成への応用、Phase Field法)
7月20日	笹川4 (第一原理計算:応用1)
7月24日	若井4 (有限要素法その1)
7月27日	笹川5 (第一原理計算:応用2)
7月31日	若井5(有限要素法その2)
8月 3日	期末試験

神谷担当講義のまとめ (試験範囲)

- コンピュータの仕組み
  コンピュータ特有の誤差と
  プログラミングにおける注意点
- 数値微分・積分
  差分法の考え方と簡単な計算
- 微分方程式の解法
  Euler法、Verlet法、Runge-kutta法の
  特徴
  二階微分方程式の解き方
- 方程式の解法
  二分法、Newton法の考え方、特徴
- 最小二乗法
  線形最小二乗法の特徴
  非線形最小化問題の解き方の
  概略、問題点・注意点
- 平滑化
  平滑化の考え方、利点と
  問題点・注意点
- フーリエ変換
  フーリエ変換の特徴と問題点
- ある実験データが与えられた時、
  どのような方法でデータ処理、
  解析ができるか
- 自分の研究で、どのような
  目的で、どのような数値解析を
  使えるか

試験対象外
- 自己無動着法
- ~~Monte Carlo法~~
- ~~KK変換~~
- ~~行列問題の解法~~

超越方程式の解法

Newton-Raphson法

$f(x) = 0$の解を求める

$f(x_0+dx) = f(x_0) + dx\, f'(x_0) \sim 0$

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

計算では $f'(x_0)$ を差分計算で置き換えられる

割線法 (セカント法、はさみうち法):

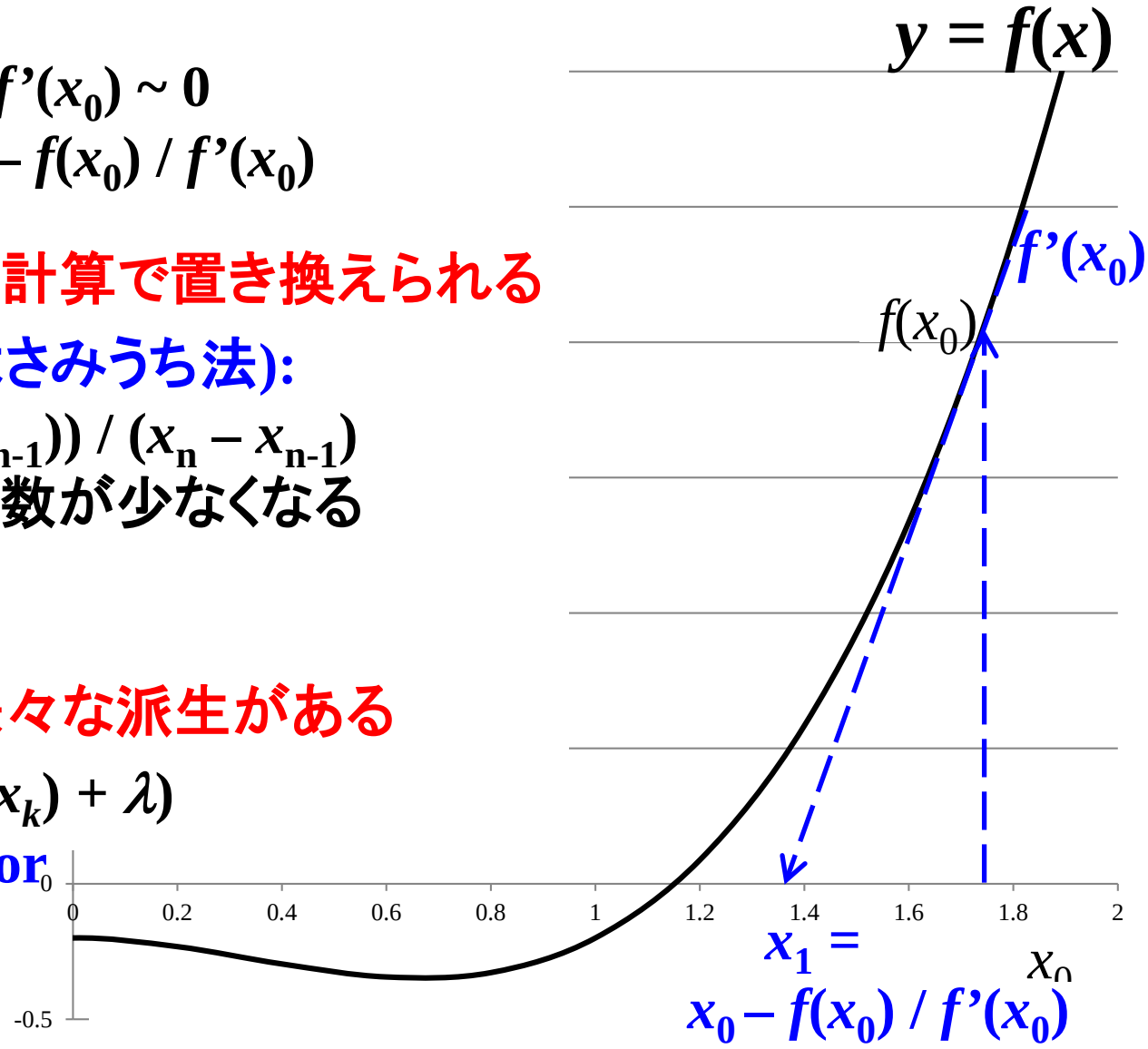
$f'(x) = (f(x_n) - f(x_{n-1})) / (x_n - x_{n-1})$

を使う。$f(x)$の計算回数が少なくなる

発散を抑える工夫で様々な派生がある

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

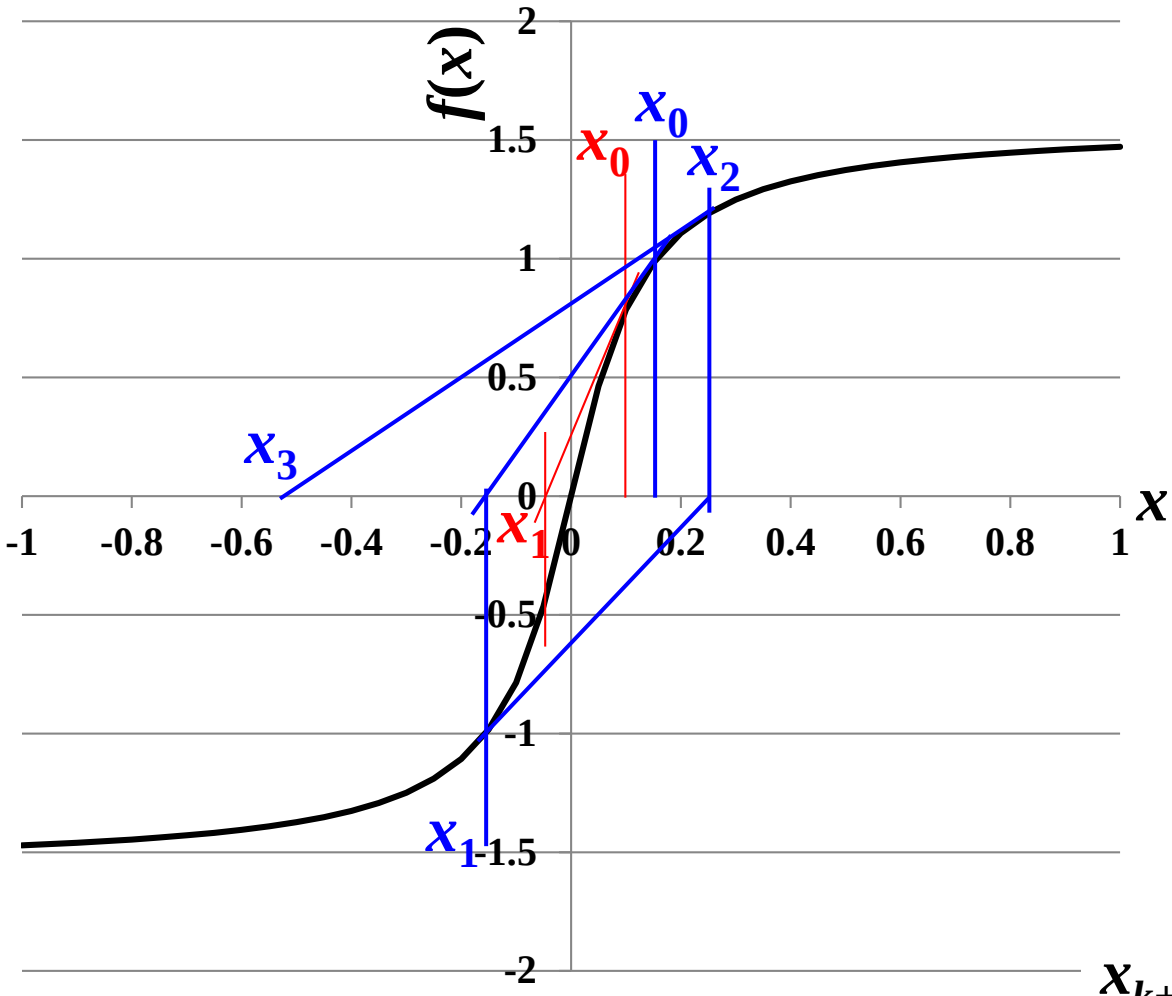
λ: Dumping Factor



Newton法が難しい例

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

初期値 $x = 0.15$



発散する場合 ($\lambda = 0$)

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

ダンピングファクターで
収束を安定化 ($\lambda = 1$)

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

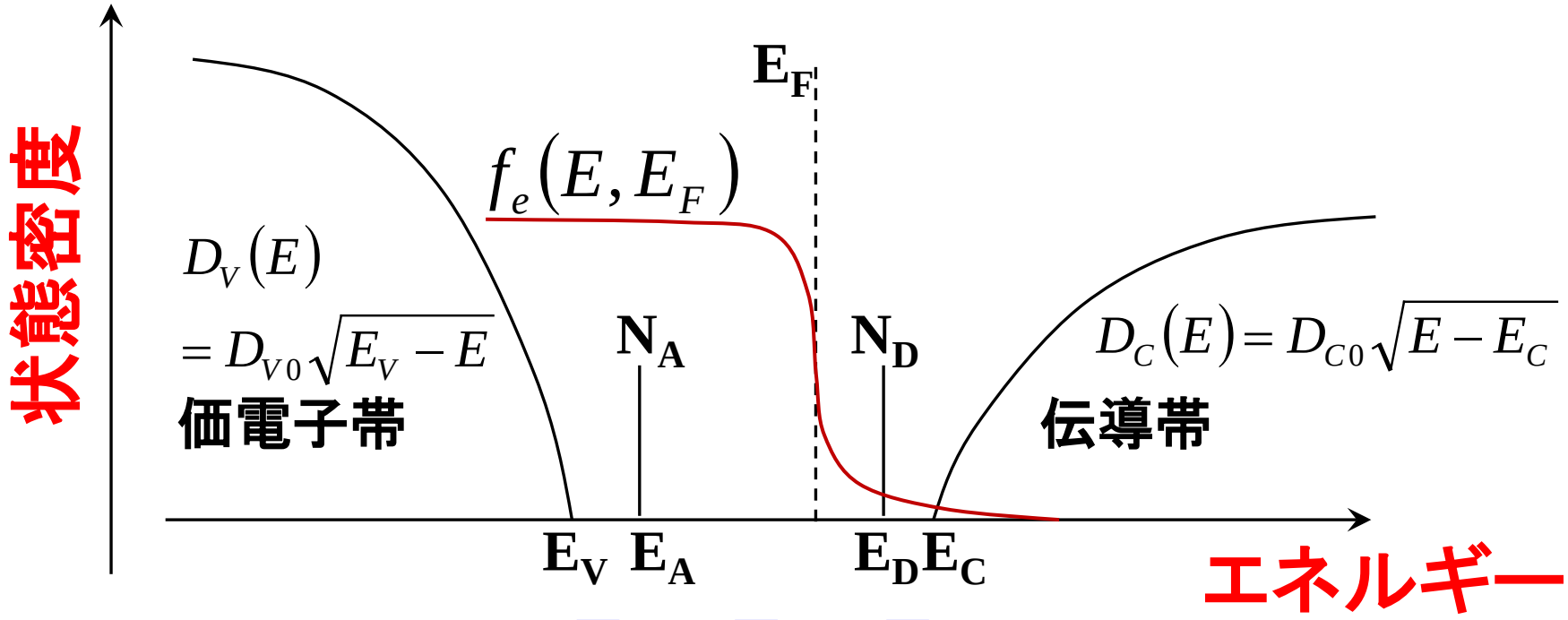
$$x_{k+1} = x_k - f(x_k) / (f'(x_k) + \lambda)$$

λ: Dumping Factor

半導体統計: 熱平衡状態での計算手順

1. パラメータ(m_e^*)を決める
2. 関連する定数(N_c, D_{c0}など)を計算する
3. 状態密度D(E)を計算する
4. 0Kで中性の状態を考え、考えているエネルギー範囲での電子数 N_e を計算する (電荷中性条件)。
5. E_F が場所によらないとして、バンド図を描く。CBM,VBMのエネルギー $E_{CBM}(x)$, $E_{VBM}(x)$が決めるべきパラメータ。
6. $E_{CBM}(x)$, $E_{VBM}(x)$ から過剰電荷密度
   $$\rho_e(x) = N_c \exp(-(E_{CBM}(x) - E_F) / k_B T)$$
   $$\rho_h(x) = N_v \exp(-(E_F - E_{VBM}(x)) / k_B T)$$
   を計算する。
7. Possisonの方程式
   $$d^2E_{CBM}(x)/dx^2 = e(-\rho_e(x)+\rho_h(x)+N_D^+(x)-N_A^-(x)) /\varepsilon$$
   を満足するように、5, 6 を自己無撞着に解く。

半導体中のフェルミ準位を求める



$$E_g = E_C - E_V$$

電荷中性条件

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

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

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

フェルミ準位: 価電子帯、アクセプターを無視できる場合

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

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

$E_C - E_F, E_D - E_F, >> k_B T$

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

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

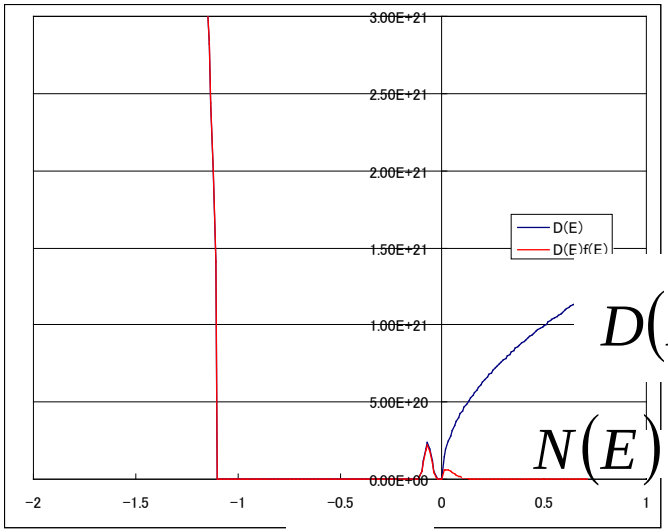
$$N_e = N_D^{+}$$

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

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

どうやってフェルミ準位を求めるか？

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	

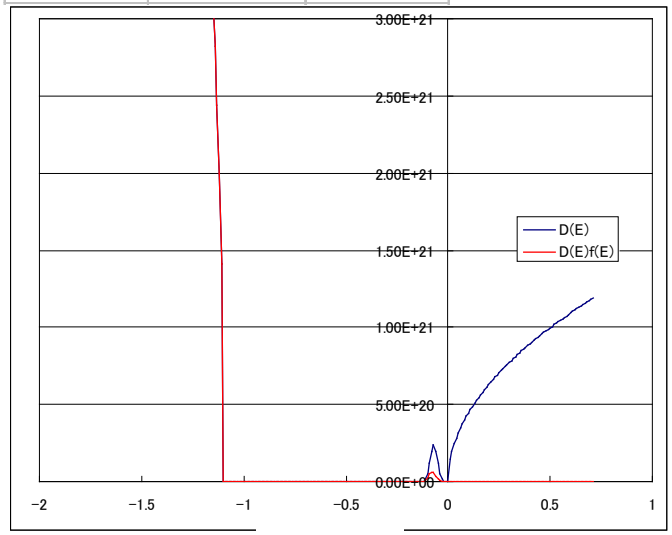


$$D(E)$$

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

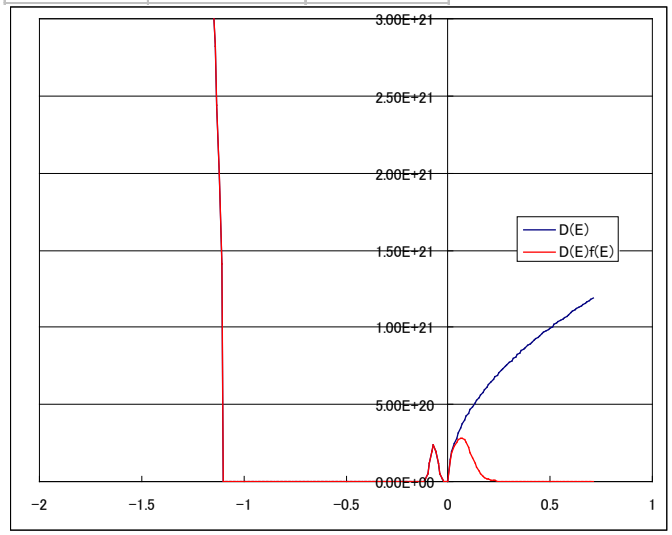
E / eV

EF=	−0.1	eV



E / eV

EF=	0.1	eV



E / eV

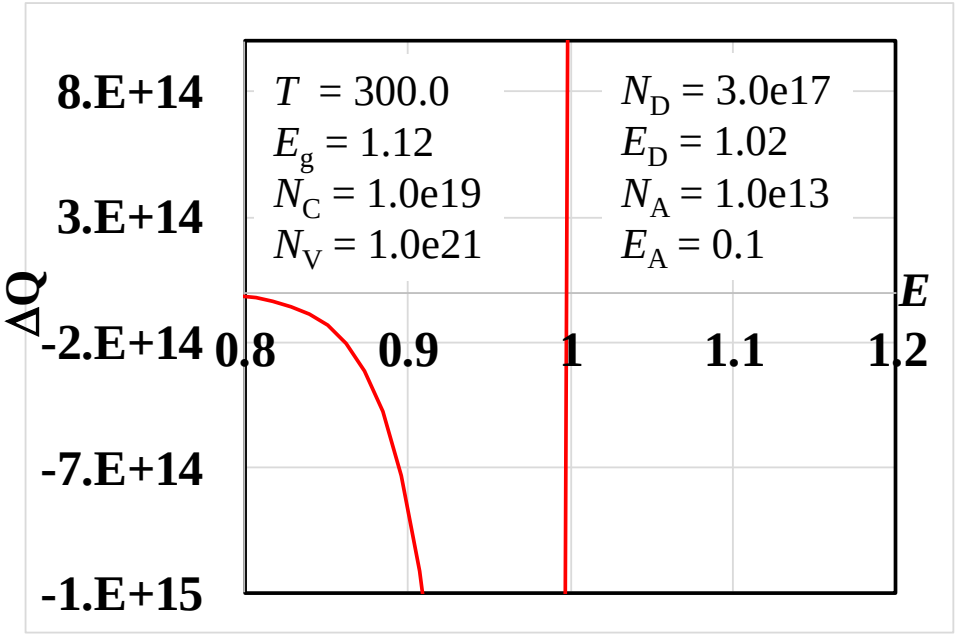
フェルミ準位の求め方: 図解

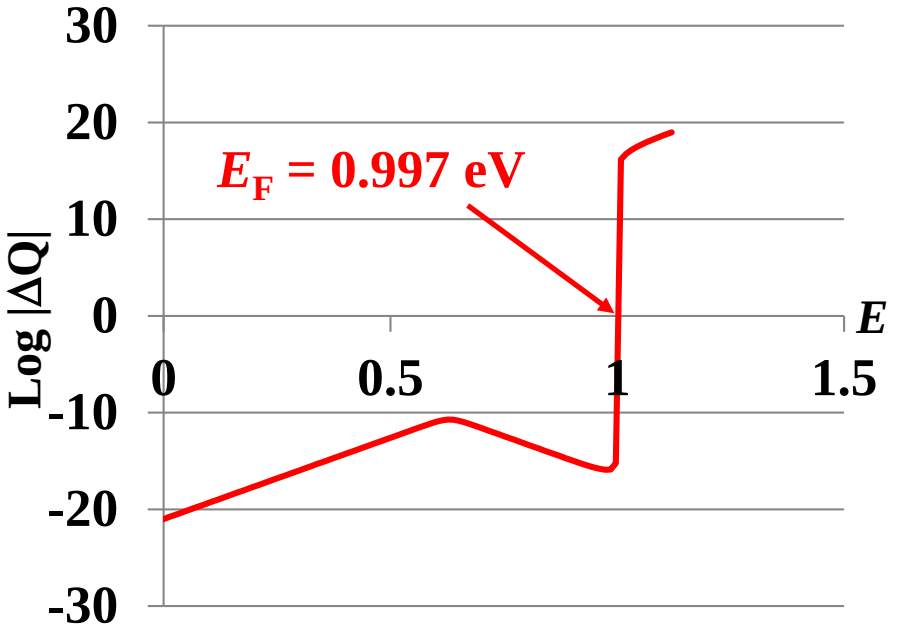
$$N_e = \int_{E_C}^{\infty} D_C(E) f_e(E, E_F) dE \qquad N_h = \int_{E_C}^{\infty} D_V(E) f_h(E, E_F) dE$$

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

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

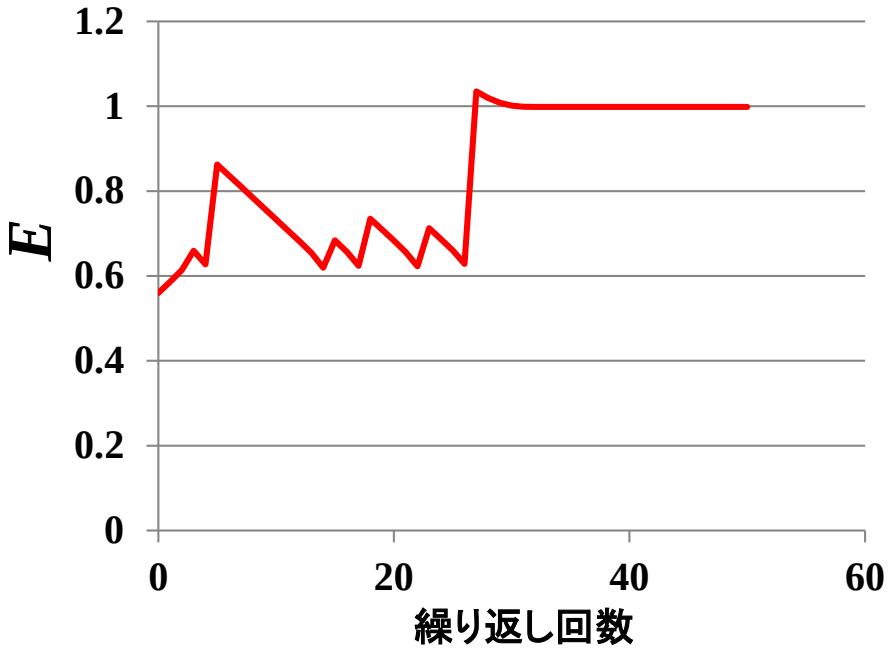
$\Delta Q = (N_A^- + N_e) - (N_D^+ + N_h)$ をE_Fに対してプロットし、ゼロ点を求める

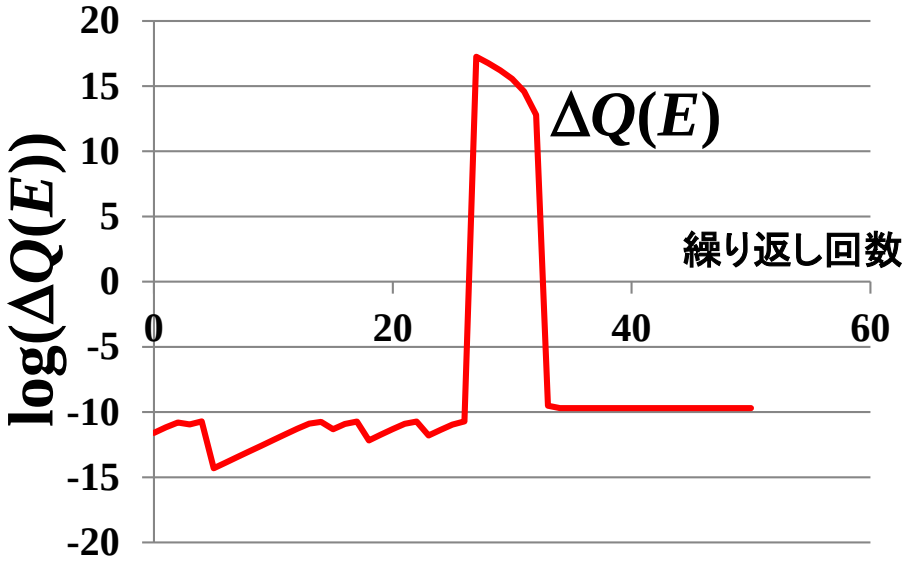




フェルミ準位の求め方: Newton-Raphson法

$E_0 = (E_V + E_C) / 2.0$ を初期値にし、
$\Delta Q = (N_A^- + N_e) - (N_D^+ + N_h)$ のゼロ点を求める





30回の繰り返し計算後

$E_F = 0.998517354556472$

$dQ = -5 \times 10^9$

単調関数方程式の解法: 二分(Bisection)法

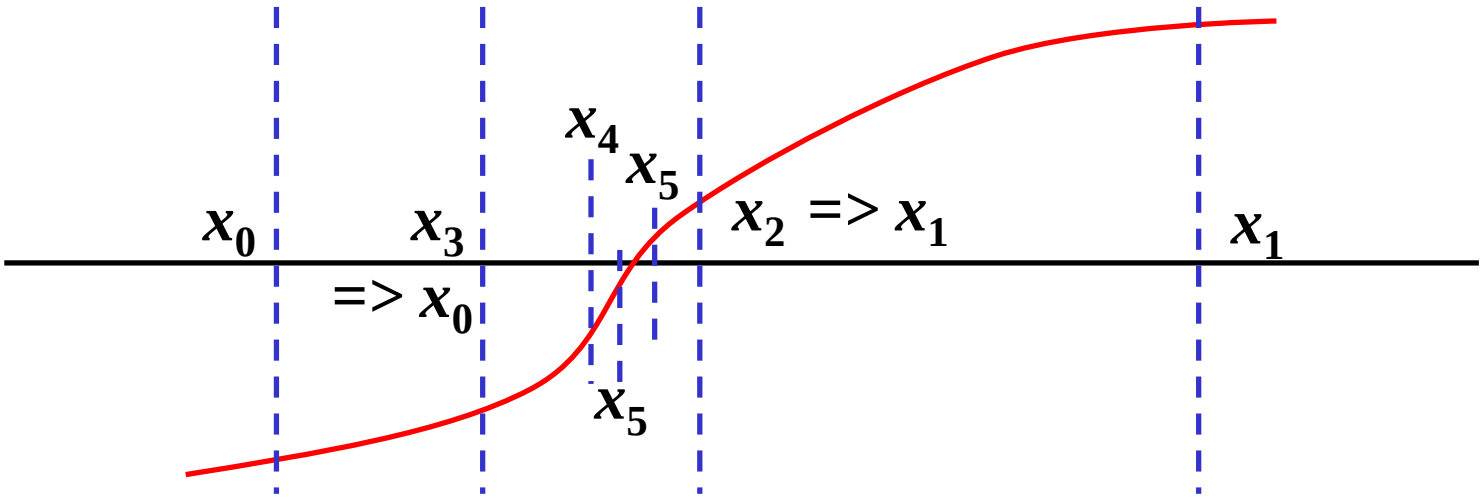
単調関数 $f(x) = 0$ の解

$f(x_0) < 0$ かつ $f(x_1) > 0$ (あるいは $f(x_0) > 0$ かつ $f(x_1) < 0$) の区間 $[x_0, x_1]$から反復的に解く

$f(x)$ が単調関数であれば、解 x はかならず $x_0 < x < x_1$

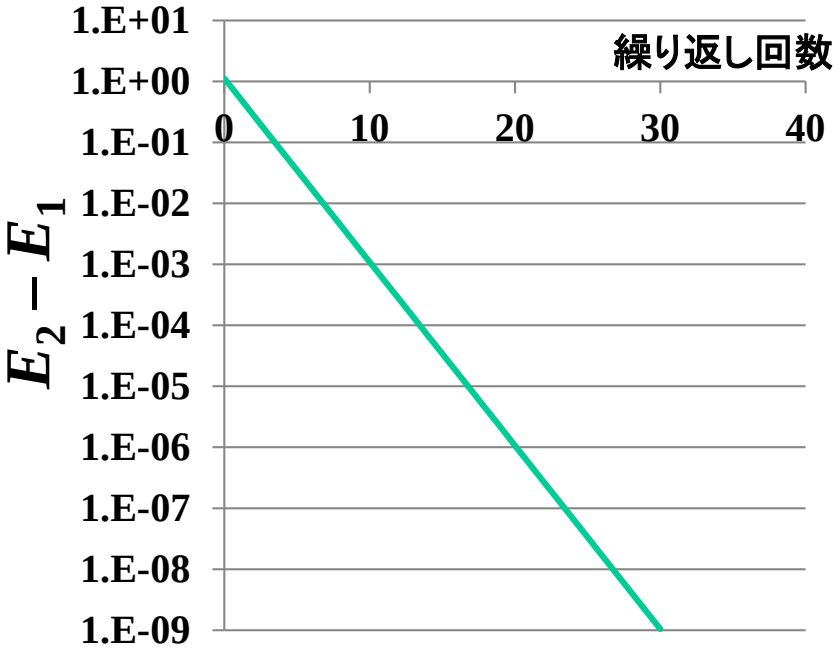
$f(x_0) < 0$ かつ $f(x_1) > 0$の場合を考える ($f(x_0) \cdot f(x_1) < 0$ で判断)。

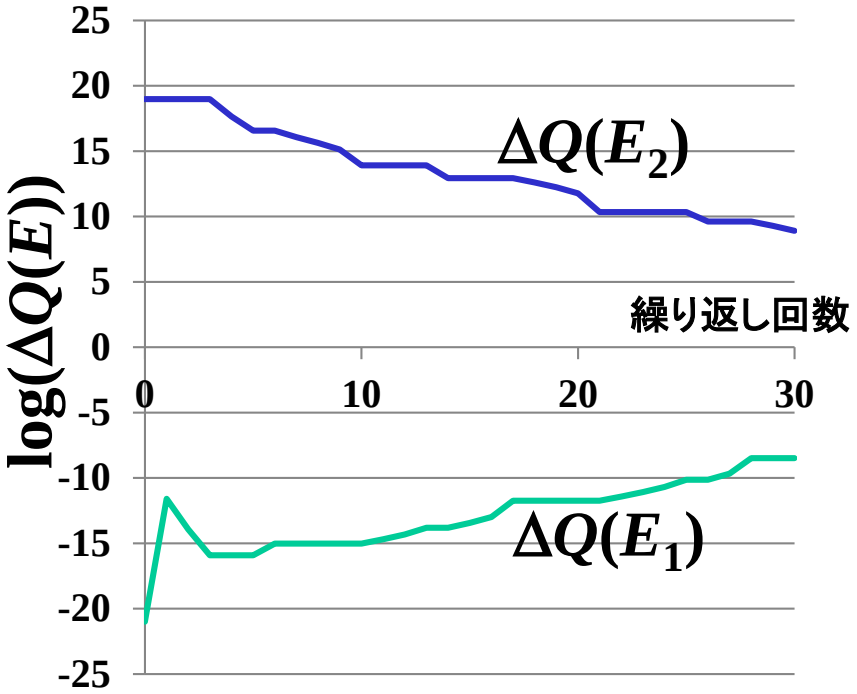
1. $x_2 = (x_0 + x_1) / 2.0$
2. $f(x_2) > 0$ ($f(x_0) \cdot f(x_2) < 0$) であれば、x_1 をx_2 で置き換える
   $f(x_2) < 0$ ($f(x_1) \cdot f(x_2) < 0$) であれば、x_0 をx_2 で置き換える
3. $|x_1 - x_0|$, $|f(x_1) - f(x_0)|$ が必要な精度以下になったら、
   解を x_2 にして反復終了
4. 1. に戻って反復



フェルミ準位の求め方: 二分法の収束過程

$[E_1, E_2] = [E_V = 0, E_C = E_g]$ を初期値にし、
$\Delta Q = (N_A^- + N_e) - (N_D^+ + N_h)$ のゼロ点を求める





30回の繰り返し計算後

$E_F = [0.9985173589, 0.9985173599]$

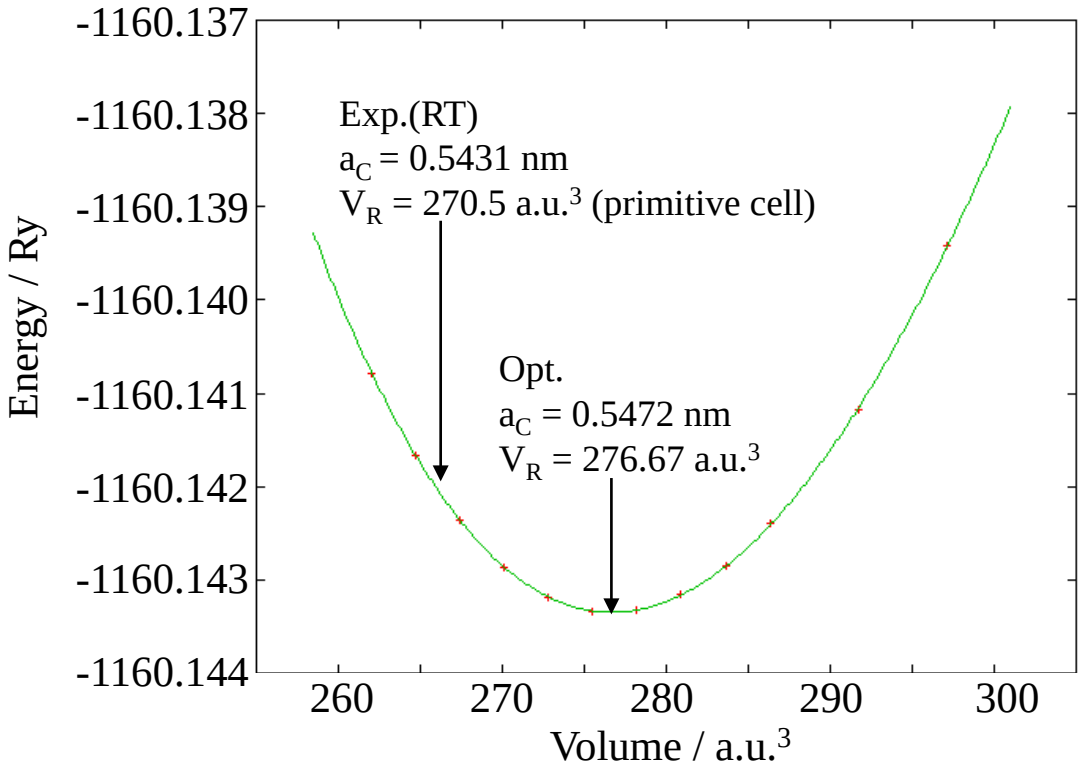
$dQ = [-3\times10^8, 8\times10^8]$

非線形多変数方程式

非線形最小化問題: 図解で安定構造を求める

第一原理計算で、格子定数を変えながら全エネルギーを計算する

例: Si

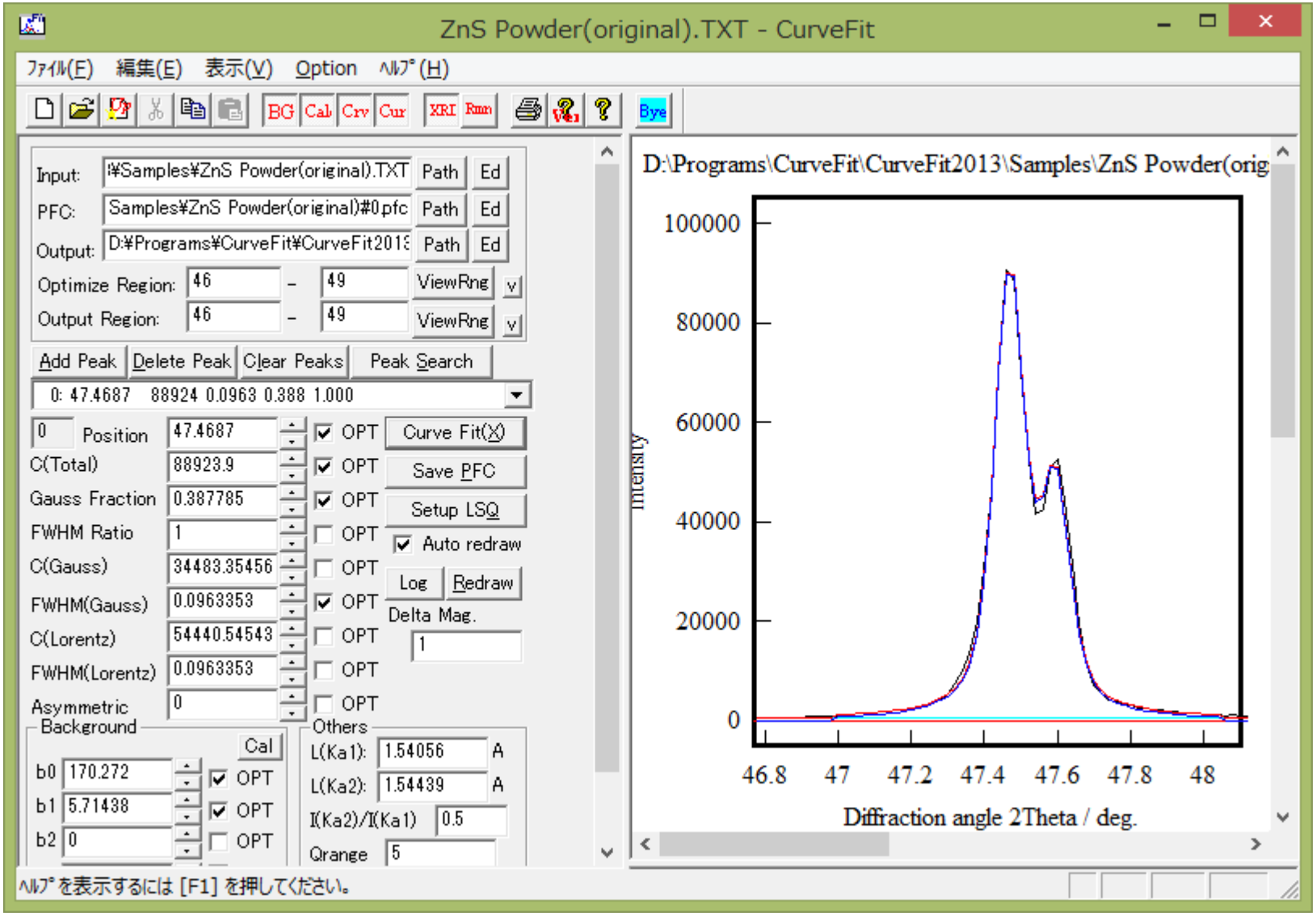


$$E = E_{\min} + 1/2 B_0 \left(V / V_0\right)^2$$

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

多変数最適化問題例: 粉末XRDのピーク分離

$K\alpha_1$と$K\alpha_2$線が 強度 2:1 で出現することを考慮



分光解析に使われるプロファイルモデル

Lorentz関数

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

w: 半値半幅

Gauss関数

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

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

Voigt関数:

Lotentz型の固有スペクトルに
他の要因のGauss型の広がりが重なる
畳み込み積分 (Convolution) であらわされる

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

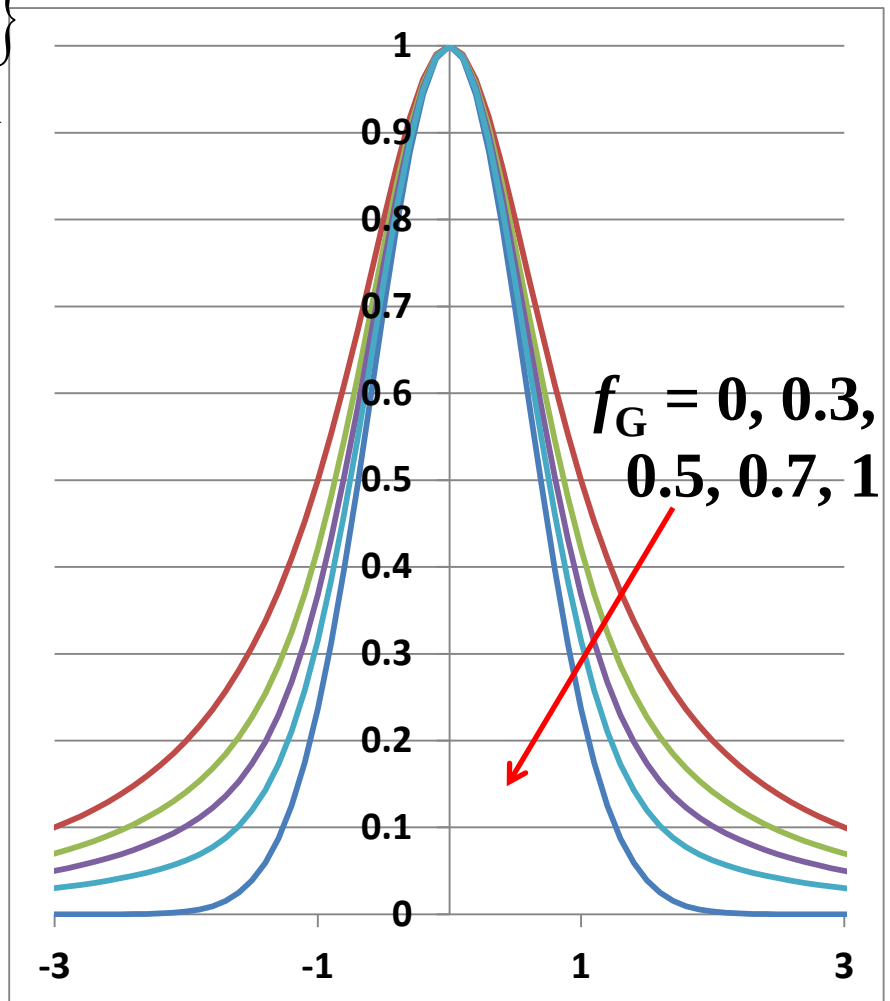
$$= \frac{a_V}{\pi} \int_{-\infty}^{\infty} \frac{\exp(-x'^2)}{a_V{}^2 + (x-x')^2} dx'$$

Pseudo-Voigt関数:

Voigt関数の簡略版

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

f_G: Gauss関数分率



多変数のNewton-Raphson法

多変数への拡張: 最小化関数 $F(x_l)$ の最小値を求める

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

繰り返し計算: $f_k(x_l + \delta x_l) \sim f_k(x_l) + \sum_{k'} \delta x_{k'} \partial f_k(x_l)/\partial x_{k'} = 0$

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

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

Hessian (ヘッセ) 行列

(ヘッセ行列の固有値をヘッシアンと呼ぶ)

Hessian行列は正定値であるとは限らない (極大値、鞍点)

＝＞ F''が降下方向を与えるとは限らない

F''を正定値行列で置き換え、発散を抑える

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

λ: Dumping Factor

最急降下法 (Steepest Descend) 法

矢部博, 工学基礎 最適化とその応用, 数理工学社 (2006)

残差関数の傾きのみから最小値を探索する。勾配法としては最も単純。

- **SD法: ベクトル $-(df / dx_i)dx_i$ 方向へ進めば S^2は小さくなる**

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

α は適当に決める。

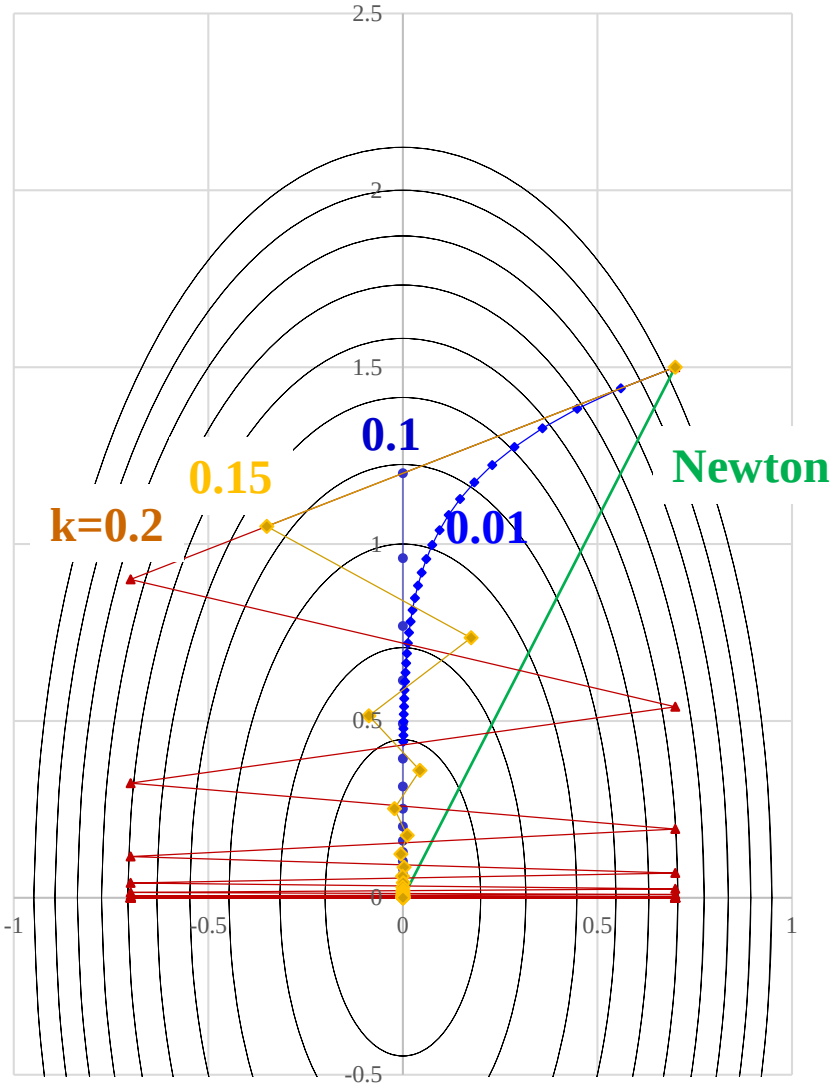
右の例:

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

- **Newton法:**
  楕円問題の場合は一度目の計算で最適値に到達

- **SD法:**
  α = 0.3: 発散 (グラフにプロットしていない)
  0.2, 0.15: 振動しながら収束
  0.1: 最初の1度目で x の最適値に到達
  0.01: 振動せず、緩やかに収束

「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」
=> 共役勾配法

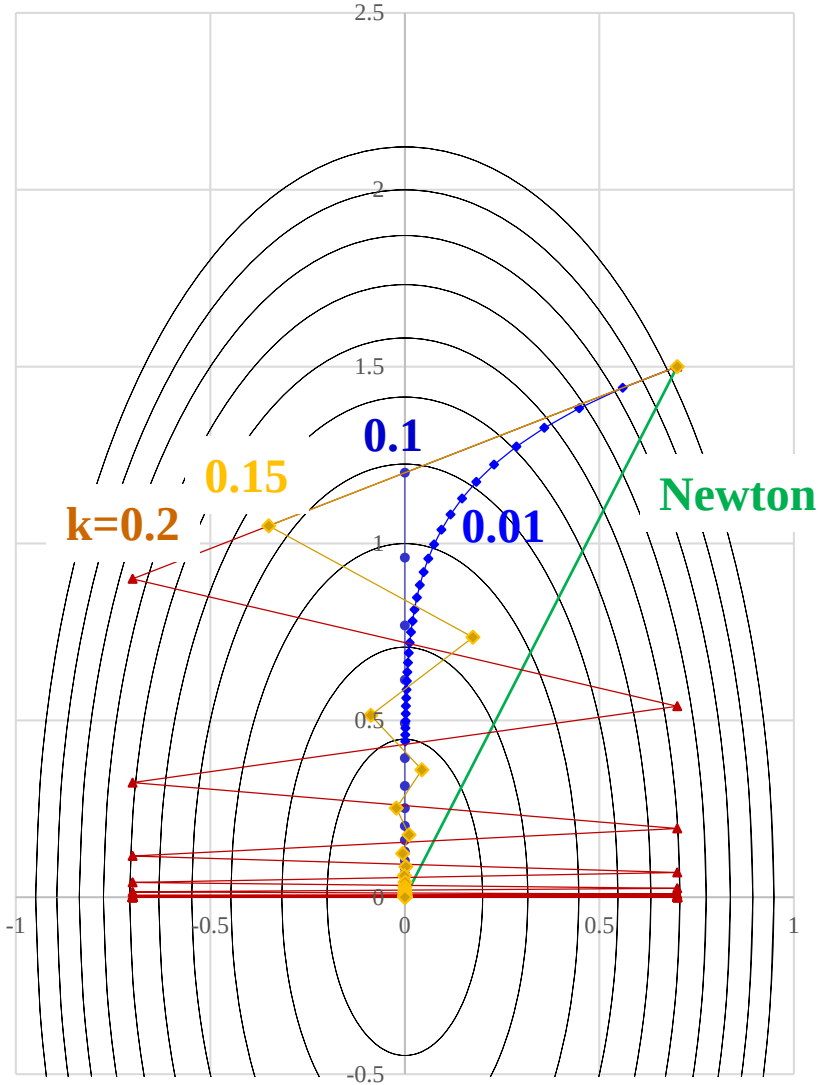


最急降下法 (Steepest Descend) 法

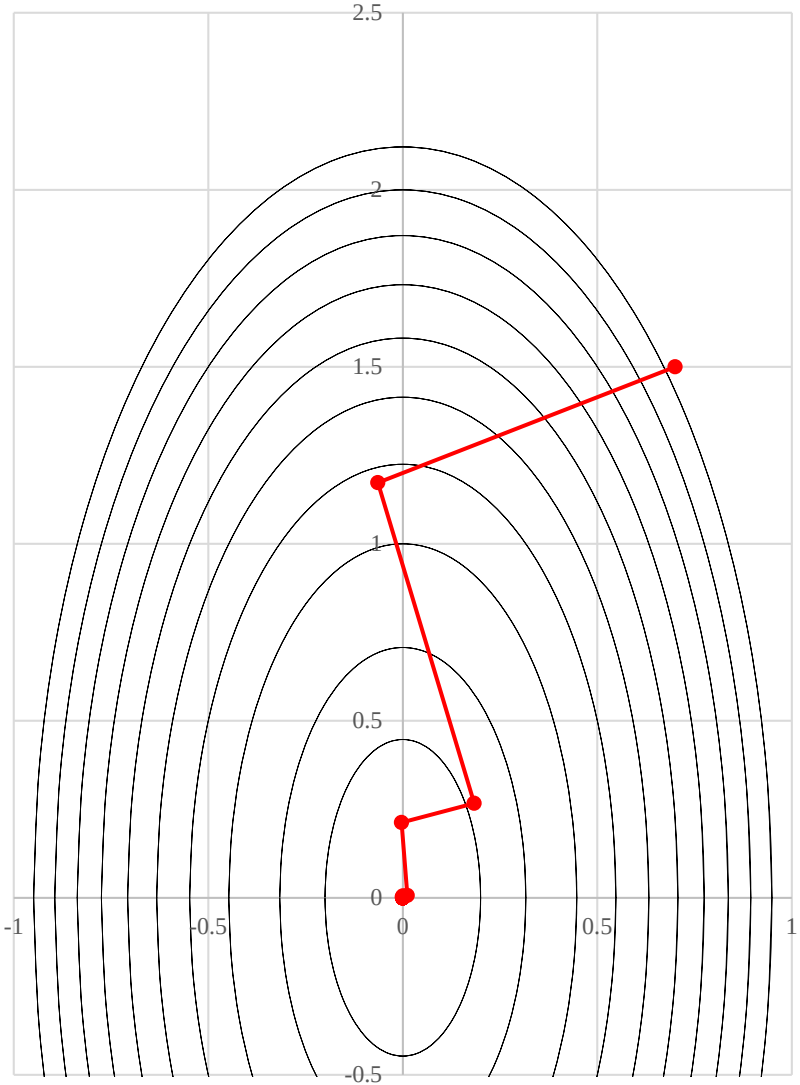
矢部博, 工学基礎 最適化とその応用, 数理工学社 (2006)

効率的な方法: 各ステップの α を直接探索法で決定する

直接探索を行わない場合



直接探索を行う場合



Deep LearningのSD法

・全Batchデータを Mini batchに分割し、
各Mini batchでSD法を実行する。

右の例:

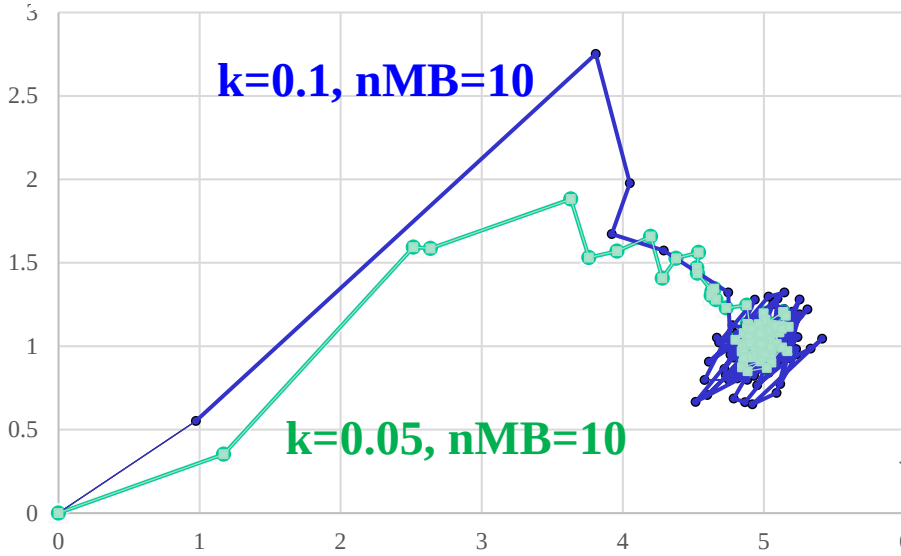
$S^2 = f(x_i) = ax_1^2 + bx_2^2$

$a = 5, b = 1$とし、乱数で1000個のデータを発生

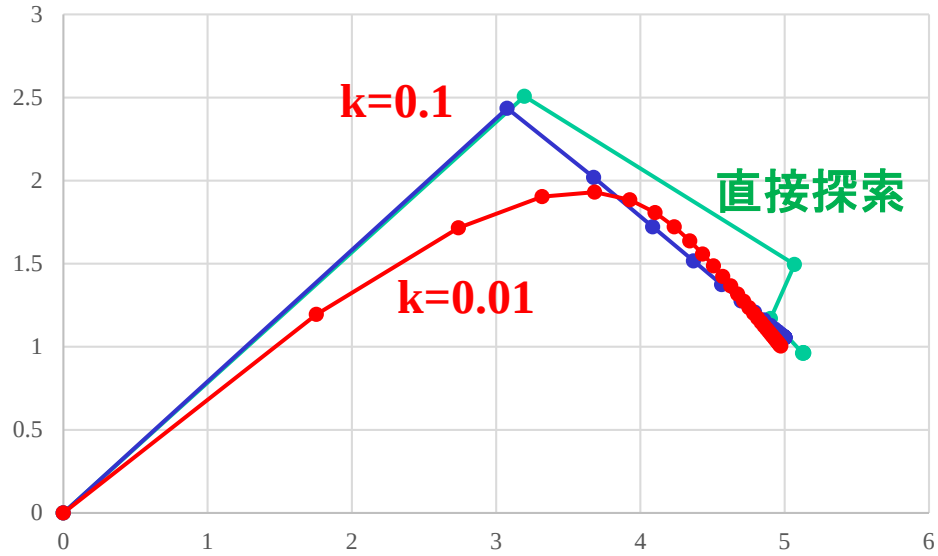
注: データは毎回作り直し

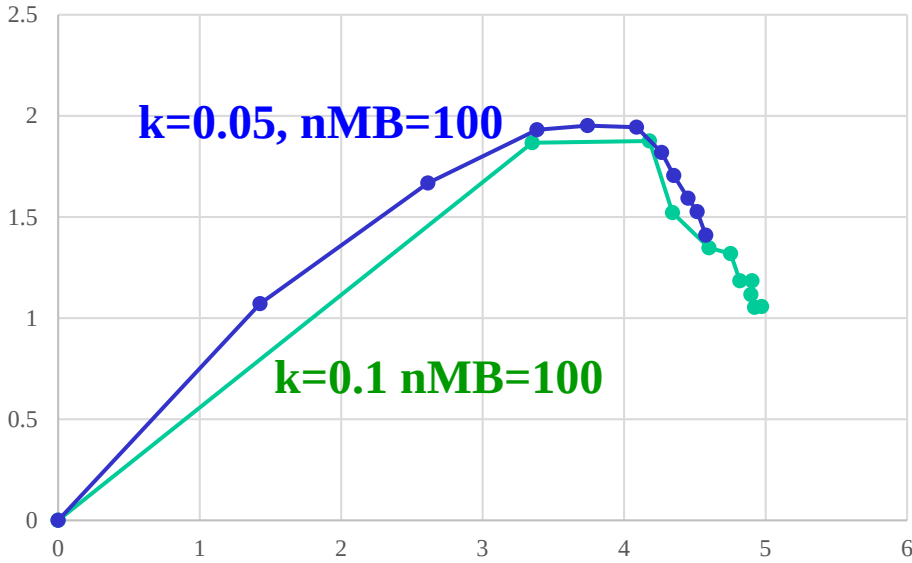
初期値 $a = 0, b = 0$

DL: SD法



SD法 (全Batchデータを繰り返して計算)





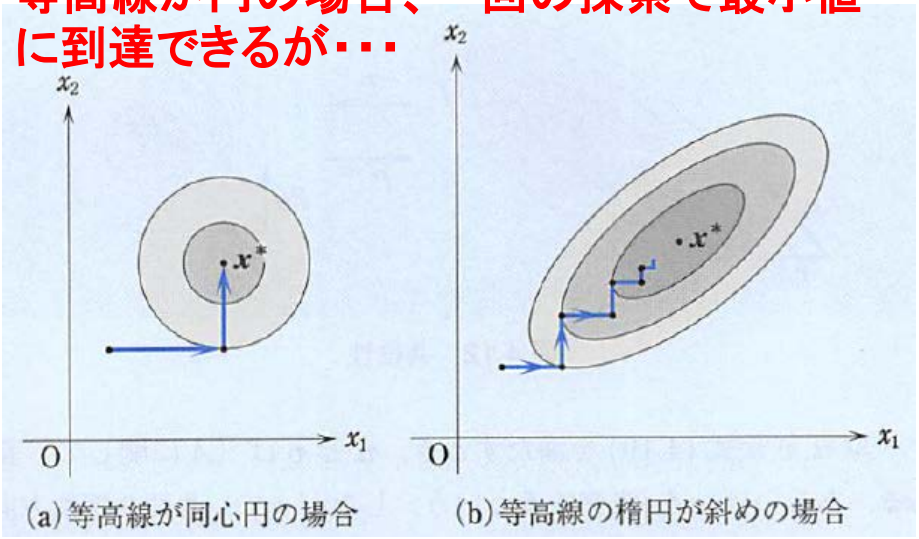
共役勾配 (Conjugate Gradient) 法

矢部博, 工学基礎 最適化とその応用, 数理工学社 (2006)

行列 A に対してベクトル u, v が $u^tAv = 0$ であるとき、u と v は互いに共役の関係にあるという

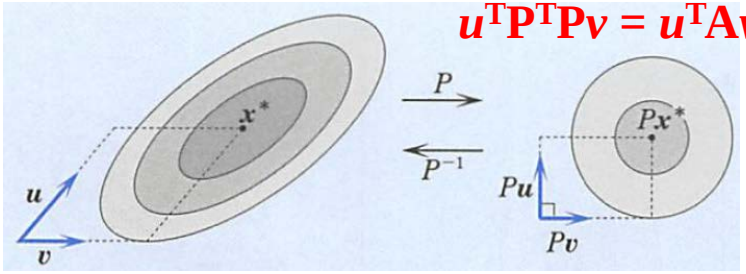
・共役な探索方向に沿って正確な直線探索を実行していけば、
有限回の反復で2次関数の最小解に到達することが期待される

等高線が円の場合、一回の探索で最小値に到達できるが・・・



共役なベクトルと楕円ー円変換

$u^TP^TPv = u^TAv = 0$



1. 初期値 x_0 を与える
2. 初期探索方向 d を再急降下方向にとる
$$\mathbf{d} = -\nabla f$$
3. 直接探索法に従ってα_kを決め、
$$x_{k+1} = x_k + \alpha_k d_k$$
を計算する
4. 探索方向を
$$\mathbf{d}_k = \mathbf{d}_{k-1} - \frac{\mathbf{d}_{k-1} \cdot \nabla f(\mathbf{x}_k)}{\mathbf{d}_{k-1} \cdot \mathbf{d}_{k-1}} \nabla f(\mathbf{x}_k)$$
に更新する
5. 3,4を繰り返して収束させる

4.では、$\mathbf{d}_k$は $\mathbf{d}_i$ (i = 1, …, k-1) のすべてに直交するため、このループは有限回しか実行できない => 時々$\mathbf{d}_k$をリセット

Marquart法

N個の変数 x_i をもつm個の関数 $f_j(x_i)$の自乗和

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

の最小値(最大値)を求める

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

と近似すると、

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

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

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

Levenberg-Marquart法

$$\boldsymbol{\delta} x=-\left(\mathbf{A}^{\mathrm{t}}\mathbf{A}+\lambda I\right)^{-1}\mathbf{A}^{\mathrm{t}}\left(f_j\right)$$ **λの最適値がよくわからない**

$$\boldsymbol{\delta} x=-\left(\mathbf{A}^{\mathrm{t}}\mathbf{A}+\lambda\,\mathrm{diag}(\mathbf{A}^{\mathrm{t}}\mathbf{A})\right)^{-1}\mathbf{A}^{\mathrm{t}}\left(f_j\right)$$ **A行列の対角和に比例させる**

単体 (Simplex) 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)

単体 (Simplex): n次元空間で (n+1) 個の頂点が作る図形

$F(x_i)$の最小値を求める

1. (n+1)個の初期値で頂点 $F(\boldsymbol{x}_i)$ (i = 1,2,･･･,n+1) を
   $F(\boldsymbol{x}_i) > F(\boldsymbol{x}_{i'})$ ($i < i'$) となるように並べ替える
2. 最大値を取る頂点 $\boldsymbol{x}_i$ 以外が作る重心を $x_G = \sum_{i=2} x_i / n$ とする

3. 直線 $\boldsymbol{x}_1$- $\boldsymbol{x}_G$ 上で新しい点を次の順に求めて F を計算する

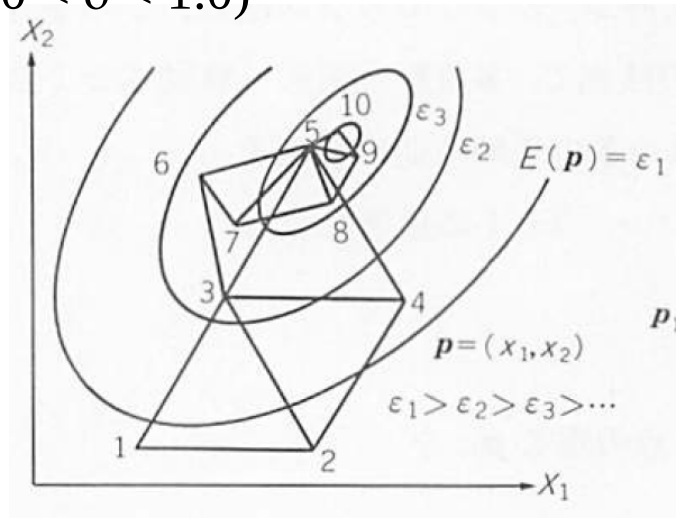
(i) 鏡映 : $\boldsymbol{x}_R = \boldsymbol{x}_1 + \alpha(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (たとえば $\alpha = 2$)

(ii) 拡大 : $\boldsymbol{x}_E = \boldsymbol{x}_1 + \beta(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (たとえば $\beta > 2$)

(iii) 縮小鏡映 : $\boldsymbol{x}_{CR} = \boldsymbol{x}_1 + \gamma(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (たとえば $1.0 < \gamma < 2.0$)

(iv) 縮小 : $\boldsymbol{x}_{CW} = \boldsymbol{x}_1 + \delta(\boldsymbol{x}_G - \boldsymbol{x}_1)$ (たとえば $0 < \delta < 1.0$)

4. (i)~(iv)のうちで最初に $F(\boldsymbol{x}) < F(\boldsymbol{x}_1)$を満たす点を
   $\boldsymbol{x}_1$と交換する。
   この操作を繰り返す



単体(Simplex)法

渡部 他 監修, Fortran 77による 数値計算ソフトウェア、丸善 (平成元年)

単体 (Simplex): n次元空間で (n+1) 個の頂点が作る図形

F(x_i)の最小値を求める

・適当な複数の初期値の組 $x_1, x_2, \cdots, x_{n+1}$ が作る超多面体から出発

・多面体を小さくしながら F(x_i) を小さくするように頂点を変える

1. F(x_i) の最大値を与える頂点を x_h, 2番目の最大値の頂点を x_0,
   最小値を与える頂点を x_1, 図心を $x_0 = \left(\sum x_i - x_h\right)/n$ とする。
2. 次の操作を行う
   - 鏡映: x_0 の x_h の反対側に最小値があると期待する
     $x_r = (1+\alpha)\, x_0 - \alpha x_h$ (たとえば $\alpha = 1$)
   - 拡張: x_0 から x_r の方向に、さらにx_r を超える
     $x_e = \gamma x_r + (1+\gamma) x_0$ (たとえば $\gamma = 2$)
   - 収縮: x_0 から x_h の方向 (単体の内側)
     $x_c = \beta x_r + (1-\beta) x_0$ (たとえば $\beta = 0.5$)
   - 縮小: 単体を x_1 の方向へ収縮
     $x_i' = (x_i + x_1) / 2$ ($i = 1,2, \cdots, n+1$)
3. 以上の操作でできた頂点で x_h, 頂点を置き換えていく

非線形最小化問題の解法

$F(x)$ の最小値(最大値)を求める

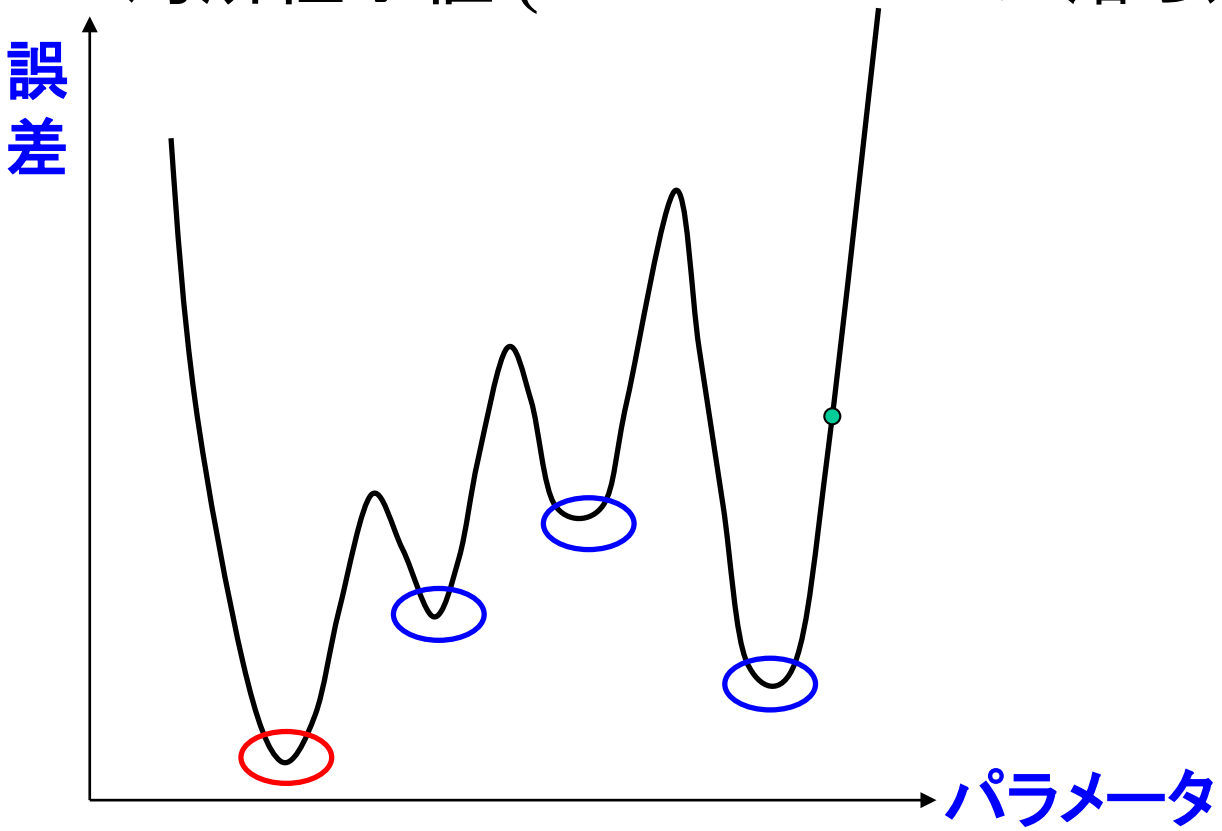
- **Newton-Raphson法:**
  二次微分行列を使って最小値を探索
- **最急降下法 (Steepest Descent):**
  一次微分から傾きの方向のみで最小値を探索
- **共役勾配 (Conjugate Gradient) 法:**
  変数の変位ベクトルの共役方向へ最小値を探索
- **Marquart法**
  $f_j(x_i)$ の一次微分の行列を使って最小値を探索

- **単体 (Simplex) 法 (Amoeba法)**
  一定のルールに従い、試行錯誤で最小値を探索

非線形 (多値) 方程式の注意

- 解が複数ある場合も
- ほとんどの場合、1度の計算で最適解を求めることは無理
  - 収束したことを確認する
  - 大域最小値を求める

⇔局所極小値 (local minimumに落ち込む)



非線形最適化アルゴリズムの傾向

	A	B
収束速度	×	○
収束安定性	○	×
安定収束範囲	○	×
使い方	第一段階	第二段階

A: 単体 (Simplex) 法

A,B: 共役勾配法 (Conjugate Gradient: CG)

B: 最急降下法 (Steepest Descent: SD)

B: Newton-Raphson法　・準Newton法

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

フーリエ変換

Fourier変換

いくつかの定義がある

Fourier変換 $F(\omega)=\int_{-\infty}^{\infty} f(t)\exp(i\omega t)dt$

Fourier逆変換 $f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\exp(-i\omega t)d\omega$

Fourier変換 $F(\omega)=\int_{-\infty}^{\infty} f(t)\exp(i2\pi ft)dt$

Fourier逆変換 $f(t)=\int_{-\infty}^{\infty} F(\omega)\exp(-i2\pi ft)d\omega$

Fourier変換の特徴

- 時系列データを周波数データに変換
- 空間系列データを波数(波長)データに変換
- 元データの原点はFTデータの全空間に拡張される
- 元の全空間データはFTデータの原点に還元される

幅 W のGauss関数のFourier変換は、幅 W^{-1} のGauss関数

Fourier変換したデータをFourier逆変換すると元のデータに戻る

Fourier級数展開

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos \frac{2\pi n}{T} t + b_n \sin \frac{2\pi n}{T} t \right)$$

$$a_n = \frac{2}{T} \int_0^T x(t) \cos \frac{2\pi n}{T} t dt$$

$$b_n = \frac{2}{T} \int_0^T x(t) \sin \frac{2\pi n}{T} t dt$$

$$x(t) = \sum_{n=-\infty}^{\infty} c_n \exp \left(i \frac{2\pi n}{T} t \right)$$

$$c_n = \frac{1}{T} \int_0^T x(t) \exp \left(-i \frac{2\pi n}{T} t \right) dt$$

リーマン・ルベーグの定理: $\lim_{n \to \infty} c_n = 0$

線形最小二乗法: 一般関数の場合

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

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

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

正弦･余弦級数展開の場合

$f_i(x)=\cos 2\pi f_i x$ **(*i* が奇数)**

$f_i(x)=\sin 2\pi f_i x$ **(*i* が偶数)**

Fourier変換 (FT)

Fourier級数展開で $T \Rightarrow \infty$ を考える

Fourier変換　$$X(f) = \int_{-\infty}^{\infty} x(t)\exp(-2\pi ft)dt$$

Fourier逆変換　$$x(t) = \int_{-\infty}^{\infty} X(f)\exp(i2\pi ft)df$$

Fourier変換の特徴

- 時系列データを周波数データに変換
- 空間系列データを波数(波長)データに変換
- 元データの原点はFTデータの全空間に拡張される
- 元の全空間データはFTデータの原点に還元される

幅 W のGauss関数のFourier変換は、幅 W^{-1} のGauss関数

Fourier変換したデータをFourier逆変換すると元のデータに戻る

離散フーリエ変換 (DFT)

$x(t)$は $[0, T^w]$以外では 0 とし、$x(0) = x(T^w)$ を仮定する

$$X(f_k) = T_s^w \sum_{j=0}^{N-1} x(t_j) \exp\left(-i2\pi f_k \cdot jT^w / N\right) \qquad T_s^w = T^w / N$$

通常、係数を含まない式を離散フーリエ変換として使う

$$y(f_k) = \sum_{j=0}^{N-1} x(t_j) \exp\left(-i2\pi kj / N\right) \qquad f_k = k / T^w$$

三角関数の計算を毎回せずに離散Fourier変換できる

$$y_k = \sum_{j=0}^{N-1} x_j w_N{}^{kj} \qquad w_N = \exp\left(-i2\pi / N\right)$$ **：回転因子**

$$\begin{aligned} w_N{}^{k+1} &= \left(\cos\left(-2\pi k / N\right) + i\sin\left(-2\pi k / N\right)\right)\left(\cos\left(-2\pi / N\right) + i\sin\left(-2\pi / N\right)\right) \\ &= \left(\cos\left(-2\pi k / N\right) w_{N,r} - \sin\left(-2\pi k / N\right) w_{N,i}\right) \\ &\quad + i\left(\cos\left(-2\pi k / N\right) w_{N,i} + \sin\left(-2\pi k / N\right) w_{N,r}\right) \\ &= \left(w_{N,r}^k w_{N,r} - w_{N,i}^k w_{N,i}\right) + i\left(w_{N,r}^k w_{N,i} + w_{N,i}^k w_{N,r}\right) \end{aligned}$$

離散フーリエ変換: 行列表現

離散Fourier変換

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & w_N^{\,1} & w_N^{\,2} & w_N^{\,N-1} \\ \vdots & w_N^{\,2} & \ddots & \vdots \\ 1 & w_N^{\,N-1} & \cdots & w_N^{\,(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}$$

逆離散Fourier変換

$$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} = \frac{1}{N} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & w_N^{\,-1} & w_N^{\,-2} & w_N^{\,-(N-1)} \\ \vdots & w_N^{\,-2} & \ddots & \vdots \\ 1 & w_N^{\,-(N-1)} & \cdots & w_N^{\,-(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix}$$

$w_N^{\,k} = w_N^{\,k \bmod N}$, $w_N^{\,k+N/2} = -w_N^{\,k}$ なので、$k = 1 \sim N/2$ までの計算だけすればよい

高速フーリエ変換 (FFT)

金谷健一, これならわかる応用数学教室, 共立出版社 (2003)

データ数は $N = 2^m$ でなければいけない
DFT (計算量 N^2) と同じ計算だが、$N\log N$ の計算量ですむ
簡単なハード回路で実装でき、多並列化が容易 (GPU)

離散Fourier変換は、$w_N^k = z$ と置き換えると、x_jを係数とする多項式になる

$$y_k = \sum_{j=0}^{N-1} x_j w_N^{kj} = \sum_{j=0}^{N-1} x_j z^j$$

$$\begin{aligned} y_k &= x_0 z^0 + x_1 z^1 + x_2 z^2 + \cdots + x_{N-1} z^{N-1} \\ &= x_0 z^0 + x_2 z^2 + \cdots + x_{N-2} z^{N-2} \\ &\quad + z(x_1 z^0 + x_2 z^2 + \cdots + x_{N-1} z^{N-2}) \end{aligned}$$

と変形すると、最後の式は、
項の数が ½ で $z_2 = z^2$ に関する多項式であることがわかる

$$y_k = \sum_{j=0}^{N/2-1} x_{2j} z_2^j + z \sum_{j=0}^{N/2-1} x_{2j+1} z_2^j$$

高速フーリエ変換 (FFT)

金谷健一, これならわかる応用数学教室, 共立出版社 (2003)

$$y_k = x_0\left(z^2\right)^0 + x_2\left(z^2\right)^1 + \cdots + x_{N-2}\left(z^2\right)^{N/2-1} + z(x_1\left(z^2\right)^0 + x_2\left(z^2\right)^1 + \cdots + x_{N-1}\left(z^2\right)^{N/2-1}) = y_{k,N/2,1} + zy_{k,N/2,2}$$

$$y_{k,N/2,1} = x_0\left(z^4\right)^0 + x_4\left(z^4\right)^1 + \cdots + x_{N-2}\left(z^4\right)^{N/4-1} + \left(z^2\right)(x_2\left(z^4\right)^0 + x_6\left(z^4\right)^1 + \cdots + x_{N-3}\left(z^4\right)^{N/4-1}) = y_{k,N/4,1} + \left(z^2\right)y_{k,N/4,3}$$

$$y_{k,N/4,2} = x_1\left(z^4\right)^0 + x_5\left(z^4\right)^1 + \cdots + x_{N-1}\left(z^4\right)^{N/4-1} + \left(z^2\right)(x_3\left(z^4\right)^0 + x_7\left(z^4\right)^1 + \cdots + x_{N-2}\left(z^4\right)^{N/4-1}) = y_{k,N/4,2} + \left(z^2\right)y_{k,N/4,4}$$

$$y_{k,N} = y_{k,N/2,1} + zy_{k,N/2,2}$$

$$y_{k,N/2,1} = y_{k,N/4,1} + z^2 y_{k,N/4,3}$$

$$y_{k,N/2,2} = y_{k,N/4,2} + z^2 y_{k,N/4,4}$$

$$y_{k,N/4,1} = y_{k,N/8,1} + z^4 y_{k,N/4,3}$$

$$y_{k,N/4,2} = y_{k,N/8,2} + z^4 y_{k,N/4,4}$$

$$y_{k,N/4,3} = y_{k,N/8,5} + z^4 y_{k,N/4,7}$$

$$y_{k,N/4,4} = y_{k,N/8,6} + z^4 y_{k,N/4,8}$$

漸化式のかたちになっているので、項数 2 のFourier変換の結果から順次、項数 $2^2, 2^3, \ldots, 2^N$のFourier変換の計算ができる

FFTでの順番入れ替え

順序数を二進数であらわす。

それぞれの段階で「奇数番目のデータを後半にずらす」操作をする
=> 順序数の右から段階数に「対応するビットが 1 のデータを後半にずらす」
=> 順序数の変換がビット反転に対応する

最初のデータの並び順 (カッコ内は順序数のビット反転)
000 (000) 001 (100) 010 (010) 011 (110) 100 (001) 101 (101) 110 (011) 111 (111)
=> 第2段階
1: 000 (111) 010 (010) 100 (001) 110 (011)
2: 001 (100) 011 (110) 101 (101) 111 (111)
=> 第3段階
1-1: 000 (000) 100 (001)
1-2: 010 (010) 110 (011)
2-1: 001 (100) 101 (101)
2-2: 011 (110) 111 (111)

=> データ使用順序
1-1-1: 000 (000)
1-1-2: 100 (001)
1-2-1: 010 (010)
1-2-2: 110 (011)
2-1-1: 001 (100)
2-1-2: 101 (101)
2-2-1: 011 (110)
2-2-2: 111 (111)

高速フーリエ変換 (FFT)

金谷健一, これならわかる応用数学教室, 共立出版社 (2003)

$$y_k = \sum_{j=0}^{N/2-1} x_{2j} z_2^{\ j} + z \sum_{j=0}^{N/2-1} x_{2j+1} z_2^{\ j}$$

漸化式のかたちになっているので、項数 2 のFourier変換の結果から順次、項数 $2^2, 2^3, \ldots, 2^N$のFourier変換の計算ができる

$$FFT_N^{\ k} = FFT_{N/2}^{\ k} + z^k FFT_{N/2}^{\ k}$$
$$FFT_N^{\ k/2+k} = FFT_{N/2}^{\ k} - z^k FFT_{N/2}^{\ k}$$

k = 0, 1, …, N/2-1

$$FFT_n^{\ k} = \sum_{j=0}^{n-1} x_{j'} z^{kj}$$

j' は 2^n ごとに間引きしたデータ

漸化式を解いて得られたFTデータの順序は、元のデータの順序とは違う

=> 元データの順序を 2進数にし、ビット反転させた値でソートしてから漸化式を解く

FFTのアルゴリズム

1. 元データの順序をビット反転により並べなおす

2. 漸化式を解いてFTデータを計算する。漸化式なので再帰アルゴリズムで解けるが、実際にはその必要はない (一般に、再帰関数を使わないほうが早い)