計算機実験 II (L4) — 最適化問題

藤堂眞治
wistaria@phys.s.u-tokyo.ac.jp

2023/12/1

講義日程 (予定)

- 全 8 回 (金曜 5 限 16:50-18:35)
  - 10 月 6 日 (金) 講義 1: 多体系の統計力学とモンテカルロ法
  - 10 月 13 日 (金) 実習 1
  - 10 月 20 日 (金) 休講 (物理学教室コロキウム)
  - 10 月 27 日 (金) 講義 2: 偏微分方程式と多体系の量子力学
  - 11 月 10 日 (金) 休講 (物理学教室コロキウム)
  - 11 月 17 日 (金) 講義 3: 少数多体系・分子動力学
  - 11 月 21 日 (火) 演習 2
  - 12 月 1 日 (金) 講義 4: 最適化問題
  - 12 月 8 日 (金) 実習 3
  - 12 月 15 日 (金) 休講 (物理学教室コロキウム)
  - 12 月 22 日 (金) 休講 (ニュートン祭)
  - 1 月 5 日 (金) 実習 4
  - 1 月 19 日 (金) 休講
  - 1 月 26 日 (金) 休講

最適化問題

- 目的関数 (コスト関数)$f(x)$ の最小値 (あるいは最大値) とその場所を求めたい
- どういう問題を解くのに使えるか？
  - 変分原理が成り立つ問題: 最小作用の原理、最小エネルギーの原理 · · ·
  - 目的関数が定義できる問題: 最小二乗法 (線形回帰、非線形回帰)、(連立) 方程式、常/偏微分方程式、機械学習 · · ·
- ほぼ全ての問題は、目的関数をうまく定義することで最適化問題に書き換えることができる
  - (一般に) 最適化問題として解くのは最終手段
  - それぞれの問題に特化したより良い方法があるときはそちらを使う

様々な最適化手法

- 最適化問題の種類
  - 連続最適化問題 ⇐ 目的関数が凸ではない場合、難しい
  - 離散最適化 (組み合せ最適化) 問題 ⇐ さらに難しい
- 真の (大局的な) 最小値 (最大値) を求めるのは難しい
- 一般的には極値を求めることしかできない
- 多次元では極小を囲い込むことができない
- 導関数を使う方法: ニュートン法、準ニュートン法、最急降下法、勾配降下法、共役勾配法 · · ·
- 使わない方法: 囲い込み法、Nelder-Mead の滑降シンプレックス法、シミュレーテッドアニーリング、量子アニーリング · · ·
- 目的関数・導関数の評価回数と収束までの反復回数のトレードオフ

ニュートン法

- 反復法により方程式 $f(x) = 0$ の解を求める
- 真の解を x_0、現在の解の候補を $x_n = x_0 + \epsilon$ とすると

$$0 = f(x_0) = f(x_0 + \epsilon - \epsilon) = f(x_n) - f'(x_n)\epsilon + O(\epsilon^2)$$

- 次の解の候補 (反復法、逐次近似法)

$$\epsilon \approx \frac{f(x_n)}{f'(x_n)} \qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- 複素変数の複素関数や多変数の場合にも自然に拡張可

多次元の場合

- $f(x) = 0$: d 次元 (非線形) 連立方程式
- x は d 次元のベクトル: $x = {}^t(x_1, x_2, \cdots, x_d)$
- $f(x)$ も d 次元のベクトル: $f(x) = {}^t(f_1(x), f_2(x), \cdots, f_d(x))$
- 真の解のまわりでの展開 ($x_n = x_0 + \epsilon$)

$$0 = f(x_0) = f(x_0 + \epsilon - \epsilon) = f(x_n) - \frac{\partial f(x_n)}{\partial x} \cdot \epsilon + O(|\epsilon|^2)$$

- ヤコビ行列 ($d \times d$): $\left[\frac{\partial f(x)}{\partial x}\right]_{ij} = \frac{\partial f_i(x)}{\partial x_j}$
- 次の解の候補: $x_{n+1} = x_n - \left[\frac{\partial f(x_n)}{\partial x}\right]^{-1} f(x_n)$

ニュートン法による最適化

- x は d 次元のベクトル: $x = {}^t(x_1, x_2, \cdots, x_d)$、目的関数 $f(x)$ はスカラー
- 勾配ベクトル: $[\nabla f(x)]_i = \dfrac{\partial f(x)}{\partial x_i}$
- 極小値 (最小値) となる条件: $\nabla f(x) = 0$
- ニュートン法で $f(x)$ を $\nabla f(x)$ で置き換えればよい
- 次の解の候補: $x_{n+1} = x_n - H^{-1}(x_n)\nabla f(x_n)$
- ヘッセ行列 (Hessian): $H_{ij}(x) = \dfrac{\partial^2 f}{\partial x_i \partial x_j}(x)$

囲い込み法 (一次元の最適化)

- 関数 $f(x)$ の極小点を求める
- $f(a) > f(b) < f(c)$ を満たす 3 点の組 $a < b < c$ の領域を狭めていく
- $[a, b]$、$[b, c]$ の広い方 (例えば後者) を b から見て、黄金比 $[1 : (1 + \sqrt{5})/2 \approx 0.382 : 0.618]$ に内分する点を x とする
  - $f(b) > f(x)$ の場合: $[b, c]$ を新しい領域にとる
  - $f(b) < f(x)$ の場合: $[a, x]$ を新しい領域にとる
- もともとの b が $[a, c]$ を $0.382 : 0.618$ に内分する点だった場合、新しい領域の幅は、どちらの場合も 0.618
- 最初の比率が黄金比からずれていたとしても、黄金比に収束
- 黄金分割法 (golden section) とも呼ばれる

最初の囲い込み

- 1 点を選び、適当な Δx を取る
- 左右に Δx 動かしてみて、関数値が小さくなる方へ動く
- どちらに進んでも関数値が大きくなる場合には、囲い込み完了
- 小さくなった場合、その方向へ再び増えるまで Δx を倍々に増やしながら進む
- 最後の 3 点で極小値を囲い込むことができる
- 囲い込み法のプログラムの例: golden_section.c

最急降下法 (steepest descent)

- 関数の微分の情報を使う
- 現在の点 x における勾配を計算

$$-\nabla f|_i = -\frac{\partial f}{\partial x_i}$$

- 坂を下る方向にそって、一次元最適化 (ニュートン法、囲い込み法)
- 動いた先の勾配の方向でさらに最適化を繰り返す
- 関数値は単調減少 $\Rightarrow$ 極小値に収束

勾配降下法 (gradient descent)

- 勾配方向に一次元最適化を行うかわりに、あらかじめ決めた一定量 (ϵ) だけ坂を下る

$$x_{n+1} = x_n - \epsilon \nabla f$$

- あらかじめ最適な ϵ を知るのは困難
- 機械学習の分野では、(なぜか) $\epsilon = 0.1$ が良いとされている
- この方法を「最急降下法」、一次元最適化を行う勾配法を「最適降下法 (optimum descent)」と呼ぶ場合も
- ϵ を自動的に調整する手法も提案されている: ADAM, Adagrad, etc

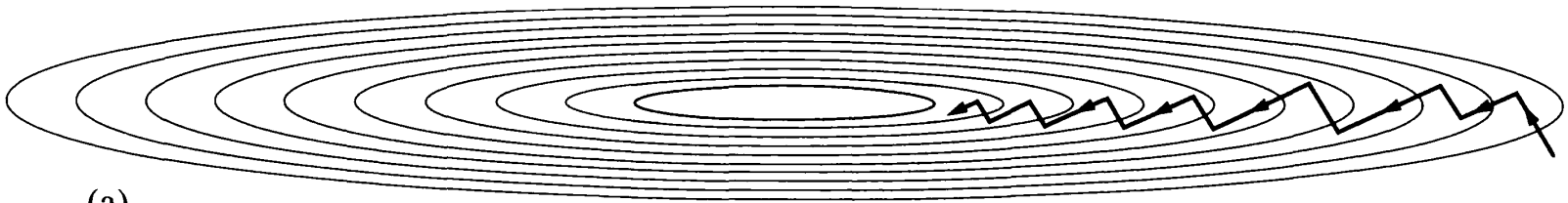
制約条件付きの場合

- 目的関数: $f(x)$
- 制約条件:
  - $g_i(x) = 0 \quad (i = 1, \cdots, m)$ (等式制約条件)
  - $h_j(x) \geq 0 \quad (j = 1, \cdots, n)$ (不等式制約条件)
- 等式制約条件の付いている場合: Lagrange の未定乗数法

$$L(x, \lambda_1, \cdots, \lambda_m) = f(x) + \sum_i \lambda_i g_i(x)$$

  を考え、$x, \lambda_1, \cdots, \lambda_m$ に関する微分が零となる点を探す
- 不等式制約条件の付いている場合: 線形計画法, ペナルティ関数法

細長い谷の場合



(a)

(b) (Press et al 1988)

共役勾配法 (conjugate gradient)

- d 次元空間の目的関数 $f(\mathbf{x})$ がある点のまわりで

$$f(\mathbf{x}) \approx c - \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T A\mathbf{x}$$

  と近似できるとする

- $\mathbf{x}$ における勾配は、連立方程式 $A\mathbf{x} = \mathbf{b}$ の「残差」の形で書ける

$$-\nabla f = \mathbf{b} - A\mathbf{x}$$

- 新しい勾配方向ではなく、それまでとは「共役な方向」に進みたい

「共役な方向」とは

- あるベクトル $\mathbf{p}$ にそった一次元の最適化が完了したとする
  - その点における $\mathbf{p}$ 方向の勾配は零。すなわち $\mathbf{p}^T(\nabla f) = 0$
  - $\mathbf{p}$ 方向の勾配の値を変化させないようにしたい
- 次に、$\mathbf{q}$ にそって、$\mathbf{x} + \epsilon\mathbf{q}$ と移動するとする。その時の勾配の変化は

$$\delta(\nabla f) = A \times (\epsilon\mathbf{q}) \sim A\mathbf{q}$$

  これが $\mathbf{p}$ に垂直であるためには

$$\mathbf{p}^T A\mathbf{q} = 0$$

- この関係が成り立つ時、$\mathbf{p}$ と $\mathbf{q}$ は「互いに共役」という

共役勾配法 (conjugate gradient)

- 初期条件: 位置 $\mathbf{x} = \mathbf{x}_0$

$$\text{勾配: } \mathbf{p} = \mathbf{p}_0 = -\nabla f(\mathbf{x}_0) = \mathbf{b} - A\mathbf{x}_0$$
$$\text{残差: } \mathbf{r} = \mathbf{r}_0 = \mathbf{p}_0$$

- 最適化 (第 n ステップ)

$$\alpha_n = \frac{\mathbf{r}_n^T \mathbf{p}_n}{\mathbf{p}_n^T A \mathbf{p}_n}$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{p}_n$$
$$\mathbf{r}_{n+1} = \mathbf{r}_n - \alpha_n A\mathbf{p}_n = \mathbf{b} - A\mathbf{x}_{n+1}$$
$$\beta_n = -\frac{\mathbf{p}_n^T A \mathbf{r}_{n+1}}{\mathbf{p}_n^T A \mathbf{p}_n}$$
$$\mathbf{p}_{n+1} = \mathbf{r}_{n+1} + \beta_n \mathbf{p}_n$$

共役勾配法 - 直線上の最適化

- 直線 $\mathbf{x} = \mathbf{x}_n + \alpha \mathbf{p}_n$ 上での最適化
- 点 $\mathbf{x}$ で $f(\mathbf{x})$ が最小値をとるためには、$\mathbf{x}$ における勾配と直線の方向ベクトル $\mathbf{p_n}$ が垂直にならなければならない

$$
\begin{aligned}
-[\nabla f(\mathbf{x})]^T \mathbf{p}_n &= (\mathbf{b} - A\mathbf{x})^T \mathbf{p}_n = (\mathbf{b} - A(\mathbf{x}_n + \alpha \mathbf{p}_n))^T \mathbf{p}_n \\
&= (\mathbf{r}_n - \alpha A \mathbf{p}_n)^T \mathbf{p}_n = 0
\end{aligned}
$$

- 最適解: $\alpha_n = \dfrac{\mathbf{r}_n^T \mathbf{p}_n}{\mathbf{p}_n^T A \mathbf{p}_n}$

共役勾配法 - 共役な方向の決め方

- 新たな最適化方向: $\mathbf{p} = \mathbf{r}_{n+1} + \beta \mathbf{p}_n$
- $\beta = 0$ とすると最急降下法と等価
- 共役勾配法では、$\mathbf{p}$ と $\mathbf{p}_n$ が共役となるように $\beta \mathbf{p}_n$ だけ補正を加える

$$\mathbf{p}_n^T A \mathbf{p} = \mathbf{p}_n^T A (\mathbf{r}_{n+1} + \beta \mathbf{p}_n) = 0$$

- β について解くと

$$\beta_n = -\frac{\mathbf{p}_n^T A \mathbf{r}_{n+1}}{\mathbf{p}_n^T A \mathbf{p}_n}$$

共役勾配法 - 共役性と直交性

- 共役性: $\{\mathbf{p}_i\}$ は自動的に互いに共役となる

$$\mathbf{p}_i^T A \mathbf{p}_j = 0 \qquad (i < j \leq n)$$

- $\mathbf{x}_n$ を通り $\mathbf{p}_0 \cdots \mathbf{p}_n$ に並行な「平面」上で $f(\mathbf{x}_{n+1})$ は最小

$$\mathbf{p}_i^T \mathbf{r}_{n+1} = 0 \qquad (i \leq n)$$

- 直交性: $\{\mathbf{r}_i\}$ は自動的に互いに直交する

$$\mathbf{r}_i^T \mathbf{r}_j = 0 \qquad (i < j \leq n+1)$$

- d 回反復すると残差は零 (実際は数値誤差により直交性がくずれる)

最適化問題として連立一次方程式の解を求める

- 行列 A を正定値対称行列とする
- 連立方程式 $A\mathbf{x} = \mathbf{b}$ の解を $\hat{\mathbf{x}}$ とすると、目的関数

$$f(\mathbf{x}) = \frac{1}{2}(\hat{\mathbf{x}} - \mathbf{x})^T A(\hat{\mathbf{x}} - \mathbf{x})$$

  は $\mathbf{x} = \hat{\mathbf{x}}$ の時、最小値 0 をとる
- $\mathbf{x}$ における目的関数の勾配は、連立方程式の「残差」の形で書ける

$$-\nabla f = A(\hat{\mathbf{x}} - \mathbf{x}) = \mathbf{b} - A\mathbf{x} \equiv \mathbf{r}$$

- $f(\mathbf{x})$ の値を計算するには真の解 $\hat{\mathbf{x}}$ が必要だが、$f(\mathbf{x})$ の値そのものではなく勾配のみがあれば良い
- 行列ベクトル積だけで計算できるので、A が疎行列の時、特に有効
  ⇒ 共役勾配法を利用

共役勾配法による一般の関数の最適化

- $f(\mathbf{x})$ は厳密な二次形式ではない
- 係数行列 A も分からない
- α_n は一次元最適化を使って反復法で求める
- β_n は、A を知らなくても計算可能

$$\beta_n = \frac{\mathbf{r}_{n+1}^T \mathbf{r}_{n+1}}{\mathbf{r}_n^T \mathbf{r}_n} \qquad \text{(Fletcher-Reeves)}$$

$$\beta_n = \frac{\mathbf{r}_{n+1}^T (\mathbf{r}_{n+1} - \mathbf{r}_n)}{\mathbf{r}_n^T \mathbf{r}_n} \qquad \text{(Polac-Ribière)}$$

実際の計算では、$\beta \leftarrow \max(0, \beta)$ として、直交性が崩れた場合には方向をリセットして、勾配の方向に下るようにする

数値微分 (差分)

- 関数のテイラー展開

$$f(x+h) = f(x) + hf'(x) + h^2 f''(x)/2 + h^3 f'''(x)/6 + \cdots$$

- 数値微分の最低次近似 (前進差分)

$$f_1(x,h) \equiv \frac{f(x+h) - f(x)}{h} = f'(x) + hf''(x)/2 + O(h^2)$$

- より高次の近似 (中心差分)

$$f_2(x,h) \equiv \frac{f(x+h) - f(x-h)}{2h} = f'(x) + h^2 f'''(x)/6 + O(h^3)$$

- 刻み h を小さくすると打ち切り誤差は減少するが、小さすぎると今度は桁落ちが大きくなる

複素差分

- 虚軸方向のテイラー展開を考える

$$f(x+ih) = f(x) + ihf'(x) - h^2 f''(x)/2 - ih^3 f'''(x)/6 + \cdots$$

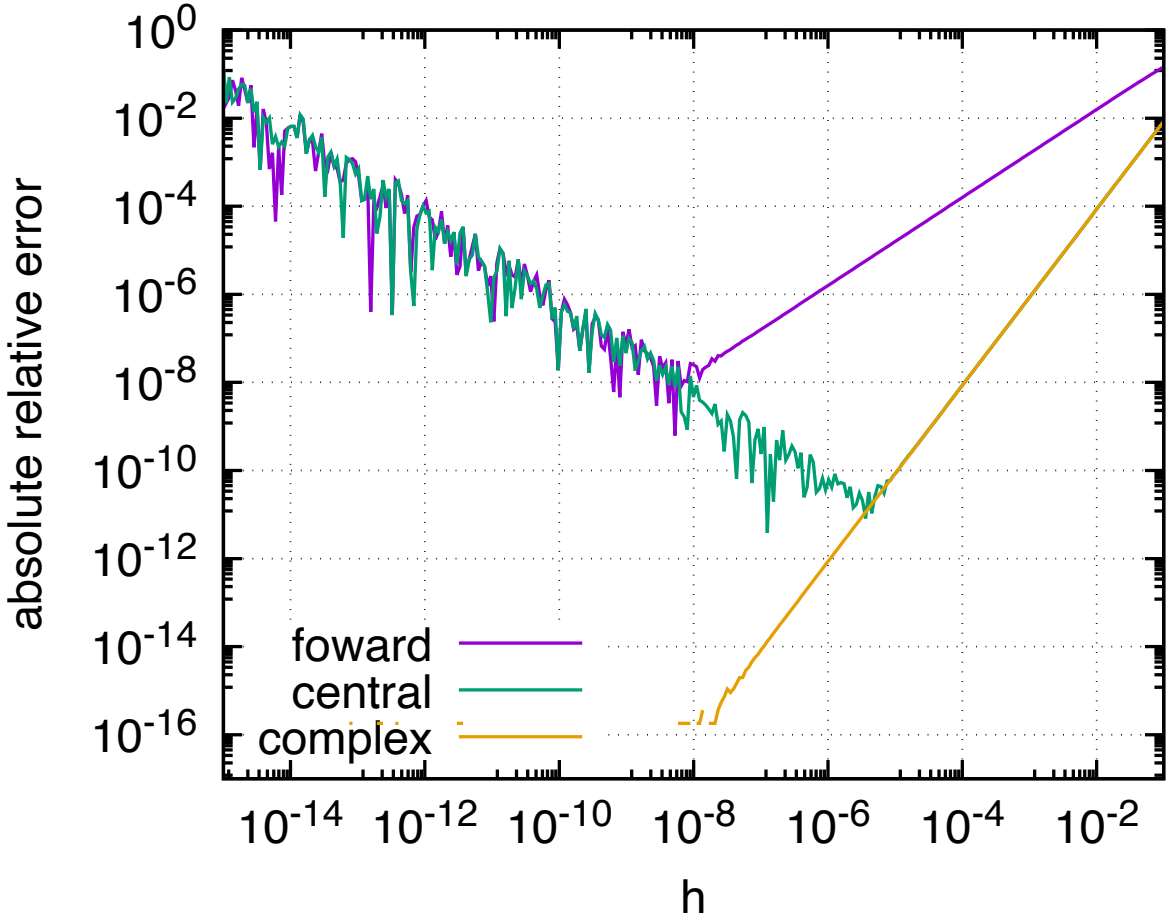
- テイラー展開の虚部を取ることで微分が求まる

$$f'(x) = \frac{\mathrm{Im}(f(x+ih))}{h} + O(h^2)$$

- 引き算がないので桁落ちに強い
- 関数の値が複素数に対して正しく計算される必要がある

前進差分・中心差分・複素差分

- 精度の比較: $\frac{d}{dx}\sin(x^2)|_{x=\pi/2}$



勾配の計算

- $f(x_1, x_2, \cdots, x_d)$ の勾配

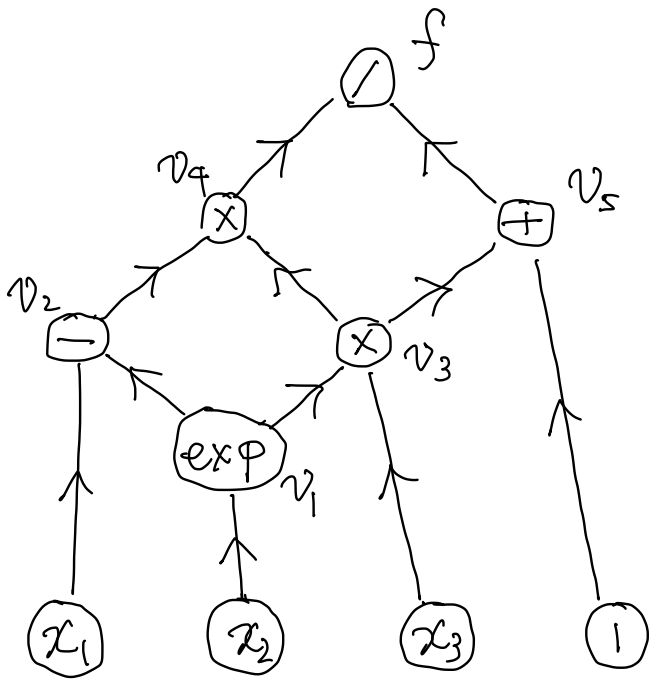
$$\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_d})$$

- 差分による計算
  - 関数 f の値を最低でも $2d$ 回評価する必要がある
  - 方向によって適切な h の値は異なる → 関数評価の回数はさらに増加
- 記号微分
  - 数式処理 (MATLAB, Mathematica 等) により勾配の表式を計算
  - f が複雑になると勾配の表式が長大に
  - 勾配の値を評価するコストも大きくなる
- 自動微分 (automatic differentiation)
  - 関数 f の値の評価に必要なコストの定数倍の手間で勾配を評価可能
  - ハイパーパラメータ h が存在しない

計算グラフ

- 任意の関数の計算過程は、入力値と基本演算からなる計算グラフ (computation graph) で表現できる

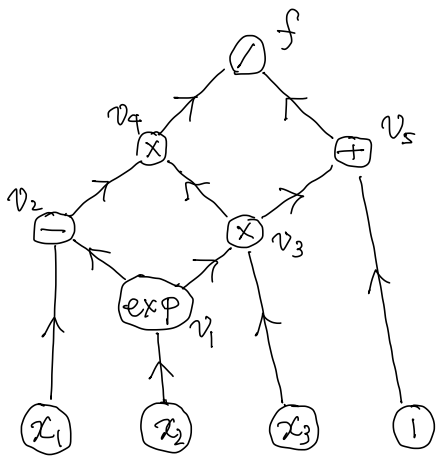
  例: $f(x_1, x_2, x_3) = \dfrac{(x_1 - \exp(x_2))x_3 \exp(x_2)}{x_3 \exp(x_2) + 1}$



[伊理・久保田 (1991)]

関数値の評価

- $f(x_1, x_2, x_3) = \dfrac{(x_1 - \exp(x_2))x_3 \exp(x_2)}{x_3 \exp(x_2) + 1}$
- $(x_1, x_2, x_3) = (2, 0, 1)$ における $f(x_1, x_2, x_3)$ の値の評価
  - $x_1 \leftarrow 2$
  - $x_2 \leftarrow 0$
  - $x_3 \leftarrow 1$
  - $v_1 \leftarrow \exp(x_2) = 1$
  - $v_2 \leftarrow x_1 - v_1 = 1$
  - $v_3 \leftarrow v_1 \times x_3 = 1$
  - $v_4 \leftarrow v_2 \times v_3 = 1$
  - $v_5 \leftarrow v_3 + 1 = 2$
  - $f \leftarrow v_4 / v_5 = 1/2$



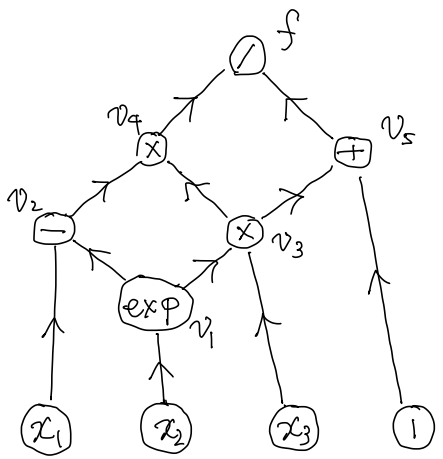
合成関数の微分

- $f(x_1, x_2, x_3)$ の x_2 に関する偏導関数の計算
- 連鎖律 (chain rule)

$$\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial v_4}\frac{\partial v_4}{\partial x_2} + \frac{\partial f}{\partial v_5}\frac{\partial v_5}{\partial x_2} = \frac{\partial f}{\partial v_4}\frac{\partial v_4}{\partial v_2}\frac{\partial v_2}{\partial x_2} + \frac{\partial f}{\partial v_4}\frac{\partial v_4}{\partial v_3}\frac{\partial v_3}{\partial x_1} + \cdots$$

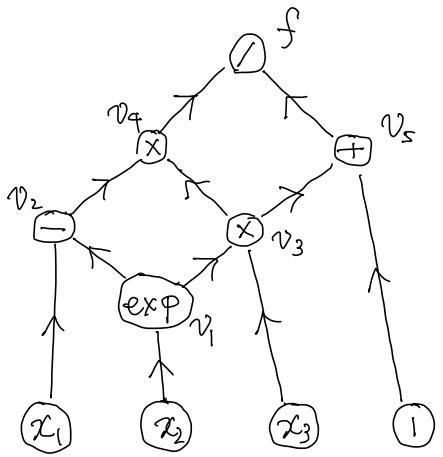
  - $\dfrac{\partial f}{\partial x_2}$ は計算経路 (全部で 7 つ) の和で書ける

- 経路の例
  - $\dfrac{\partial f}{\partial v_4}\dfrac{\partial v_4}{\partial v_2}\dfrac{\partial v_2}{\partial v_1}\dfrac{\partial v_1}{\partial x_2}\dfrac{\partial x_2}{\partial x_2}$
  - $\dfrac{\partial f}{\partial v_5}\dfrac{\partial v_5}{\partial v_3}\dfrac{\partial v_3}{\partial x_3}\dfrac{\partial x_3}{\partial x_2}(=0)$



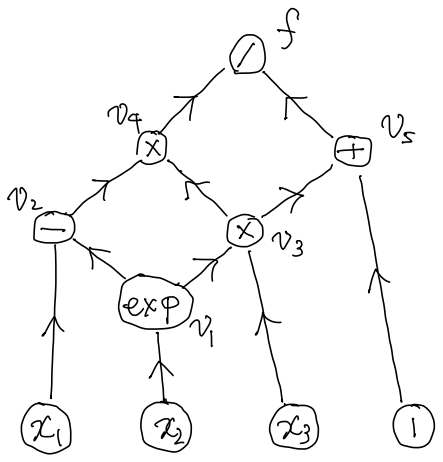
自動微分

- フォワードモード (forward mode)・ボトムアップ型 (bottom-up) 計算
- 計算グラフを下から順番に計算していく
  - $\dfrac{\partial x_1}{\partial x_2} = 0,\ \dfrac{\partial x_2}{\partial x_2} = 1,\ \dfrac{\partial x_3}{\partial x_2} = 0,\ \dfrac{\partial 1}{\partial x_2} = 0$
  - $\dfrac{\partial v_1}{\partial x_2} = \dfrac{\partial}{\partial x_2}\exp(x_2) = \exp(0) = 1$
  - $\dfrac{\partial v_2}{\partial x_2} = \dfrac{\partial x_1}{\partial x_2} - \dfrac{\partial v_1}{\partial x_2} = 0 - 1 = -1$
  - $\dfrac{\partial v_3}{\partial x_2} = \dfrac{\partial v_1}{\partial x_2} x_3 + v_1 \dfrac{\partial x_3}{\partial x_2} = 1 \times 1 + 1 \times 0 = 1$
  - $\dfrac{\partial v_4}{\partial x_2} = \dfrac{\partial v_2}{\partial x_2} v_3 + v_2 \dfrac{\partial v_3}{\partial x_2} = (-1) \times 1 + 1 \times 1 = 0$
  - $\dfrac{\partial v_5}{\partial x_2} = \dfrac{\partial v_3}{\partial x_2} + \dfrac{\partial 1}{\partial x_2} = 1 + 0 = 1$
  - $\dfrac{\partial f}{\partial x_2} = \dfrac{\partial}{\partial x_2}\dfrac{v_4}{v_5} = \dfrac{\frac{\partial v_4}{\partial x_2} v_5 - v_4 \frac{\partial v_5}{\partial x_2}}{(v_5)^2} = \dfrac{0 \times 2 - 1 \times 1}{2^2} = -1/4$



テイラー展開によるフォワードモードの計算

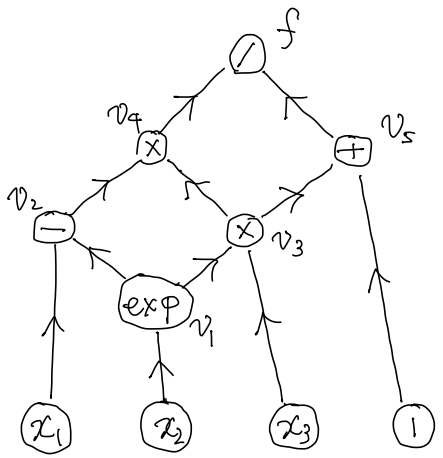
- $x_2 = 0 + \epsilon$, $x_1 = 2$, $x_3 = 1$ と表す
- 途中の計算を ϵ の 1 次まで保存しながら計算
  - $v_1 \leftarrow \exp(x_2) = \exp(0 + \epsilon) \approx 1 + \epsilon$
  - $v_2 \leftarrow x_1 - v_1 = 2 - (1 + \epsilon) = 1 - \epsilon$
  - $v_3 \leftarrow v_1 \times x_3 = (1 + \epsilon) \times 1 = 1 + \epsilon$
  - $v_4 \leftarrow v_2 \times v_3 = (1 - \epsilon) \times (1 + \epsilon) \approx 1$
  - $v_5 \leftarrow v_3 + 1 = (1 + \epsilon) + 1 = 2 + \epsilon$
  - $f \leftarrow v_4 / v_5 = 1/(2 + \epsilon) \approx \frac{1}{2} - \frac{1}{4}\epsilon$



- ϵ の係数が微分の値
- それぞれの基本演算に対するテイラー展開係数の計算式を用意しておけばよい　例)
  - $(a + b\epsilon) \times (c + d\epsilon) \approx ac + (ad + bc)\epsilon$
  - $\exp(a + b\epsilon) \approx \exp(a) + \exp(a)b\epsilon$
- 1 つの独立変数に関する偏導関数の計算は、関数値 f の計算コストと同じオーダー → 勾配を計算するにはその d 倍の計算量が必要

自動微分 (リバースモード)

- リバースモード (reverse mode)・トップダウン型 (top-down) 計算
- まず、計算グラフの下から順に $v_1, v_2, \cdots$ の値、および各基本演算における微分係数をもとめておく
  - $v_1 \leftarrow \exp(x_2) = 1,\ d_{v_1,x_2} \leftarrow \frac{\partial v_1}{\partial x_2} = \exp(x_2) = 1$
  - $v_2 \leftarrow x_1 - v_1 = 1,\ d_{v_2,x_1} \leftarrow \frac{\partial v_2}{\partial x_1} = 1,\ d_{v_2,v_1} \leftarrow \frac{\partial v_2}{\partial v_1} = -v_1 = -1,$
  - $v_3 \leftarrow v_1 \times x_3 = 1,\ d_{v_3,v_1} \leftarrow \frac{\partial v_3}{\partial v_1} = x_3 = 1,\ d_{v_3,x_3} \leftarrow \frac{\partial v_3}{\partial x_3} = v_1 = 1$
  - $v_4 \leftarrow v_2 \times v_3 = 1$
  - $d_{v_4,v_2} \leftarrow \frac{\partial v_4}{\partial v_2} = v_3 = 1$
  - $d_{v_4,v_3} \leftarrow \frac{\partial v_4}{\partial v_3} = v_2 = 1$
  - $v_5 \leftarrow v_3 + 1 = 2$
  - $d_{v_5,v_3} \leftarrow \frac{\partial v_5}{\partial v_3} = 1$
  - $f \leftarrow v_4/v_5 = 1/2$
  - $d_{f,v_4} \leftarrow \frac{\partial f}{\partial v_4} = \frac{1}{v_5} = \frac{1}{2}$
  - $d_{f,v_5} \leftarrow \frac{\partial f}{\partial v_5} = -\frac{v_4}{(v_5)^2} = -\frac{1}{4}$



自動微分 (リバースモード)

- 次に計算グラフの上から順に連鎖律を適用
  - $\frac{\partial f}{\partial v_5} = d_{f,v_5} = -\frac{1}{4}$
  - $\frac{\partial f}{\partial v_4} = d_{f,v_4} = \frac{1}{2}$
  - $\frac{\partial f}{\partial v_3} = \frac{\partial f}{\partial v_4}\frac{\partial v_4}{\partial v_3} + \frac{\partial f}{\partial v_5}\frac{\partial v_5}{\partial v_3} = d_{f,v_4}d_{v_4,v_3} + d_{f,v_5}d_{v_5,v_3} = \frac{1}{2}\cdot 1 - \frac{1}{4}\cdot 1 = \frac{1}{4}$
  - $\frac{\partial f}{\partial v_2} = \frac{\partial f}{\partial v_4}\frac{\partial v_4}{\partial v_2} = d_{f,v_4}d_{v_4,v_2} = \frac{1}{2}\cdot 1 = \frac{1}{2}$
  - $\frac{\partial f}{\partial v_1} = \frac{\partial f}{\partial v_2}\frac{\partial v_2}{\partial v_1} + \frac{\partial f}{\partial v_3}\frac{\partial v_3}{\partial v_1} = d_{f,v_2}d_{v_2,v_1} + d_{f,v_3}d_{v_3,v_1} = \frac{1}{2}\cdot(-1) + \frac{1}{4}\cdot 1 = -\frac{1}{4}$
  - $\frac{\partial f}{\partial x_3} = \frac{\partial f}{\partial v_3}\frac{\partial v_3}{\partial x_3} = \frac{1}{4}\cdot 1 = \frac{1}{4}$
  - $\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial v_1}\frac{\partial v_1}{\partial x_2} = -\frac{1}{4}\cdot 1 = -\frac{1}{4}$
  - $\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial v_2}\frac{\partial v_2}{\partial x_1} = d_{f,v_2}d_{v_2,x_1} = \frac{1}{2}\cdot 1 = \frac{1}{2}$
- 計算の過程で、全ての独立・中間変数に関する偏導関数が自動的に求まる
- 計算量は f の値を計算するコストの定数倍

自動微分

- 独立変数の次元が d, 関数値の次元が r のとき
  - $d \ll r$: フォワードモードが効率的
  - $d \gg r$: リバースモードが効率的
- $r = 1$ のとき: $f = f(x_1, x_2, \cdots x_d)$
  - ある方向 $\mathbf{p}$ に沿った微分 → フォワードモードにより効率的に計算可
  - ヘッセ行列 $H_{ij} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$ とベクトル $\mathbf{p}$ の積 → リバースモードにより効率的に計算可
- 深層学習 (ニューラルネットワーク) における最適化
  - バックプロパゲーションによる勾配の計算: 自動微分 (リバースモード) と等価
- 自動微分ライブラリ
  - boost::autodiff (C++, forward のみ), TensorFlow (Python), PyTorch (Python), Zygote (Julia) など様々なライブラリが利用可

Nelder-Mead の滑降シンプレックス法

- 関数値のみ。導関数の情報を必要としない
- プログラミングが簡単
- 収束は遅いが、安定に極小値が求まる
- $d+1$ 個の頂点からなる d 次元の超多面体 (シンプレックス) を変形しながら、極小値を探す
  - 2 次元: 三角形
  - 3 次元: 四面体
- 別名「アメーバ法」

Nelder-Mead の滑降シンプレックス法

- $d+1$ 個の点 $x_0, x_1, \cdots, x_d$ は $f(x_0) \leq f(x_1) \leq \cdots \leq f(x_d)$ の順に並べられているとする
- 最大値を取る点 x_d を除く d 点の重心を x_g とする
- Nelder-Mead 法では以下のステップを繰り返す
  - x_d を x_g に関する対称な点 x_r に移動 (反射)

$$x_r = x_g + (x_g - x_d)$$

  - $f(x_r)$ が $f(x_0)$ よりも小さい場合、さらに先に進む (拡大)

$$x_e = x_g + 2(x_r - x_g)$$

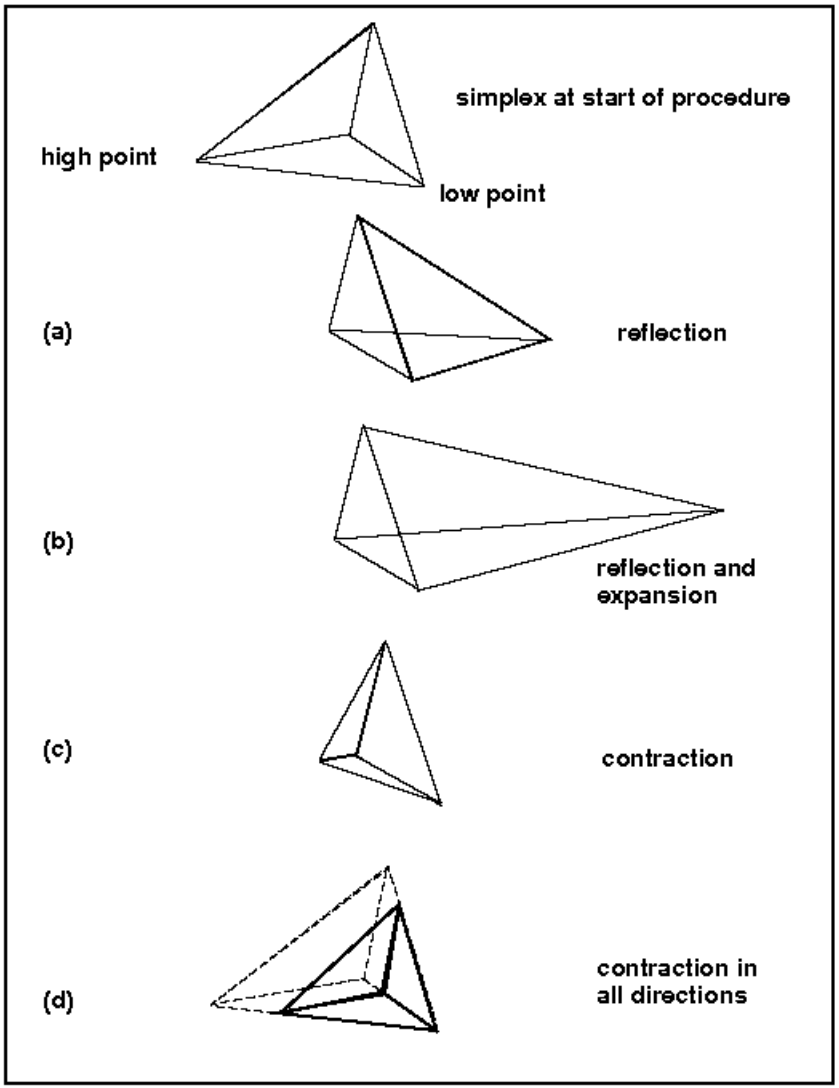
  - $f(x_r)$ が $f(x_{d-1})$ よりもまだ大きい場合には、x_d を x_g に近づける (縮小)

$$x_c = x_g + (x_d - x_g)/2$$

  - $f(x_c)$ が $f(x_d)$ よりまだ大きい場合には、x_0 以外の点を x_0 に一様に近づける (収縮)

$$x_i \leftarrow x_0 + (x_i - x_0)/2 \qquad (i = 1 \cdots d)$$

Nelder-Mead の滑降シンプレックス法



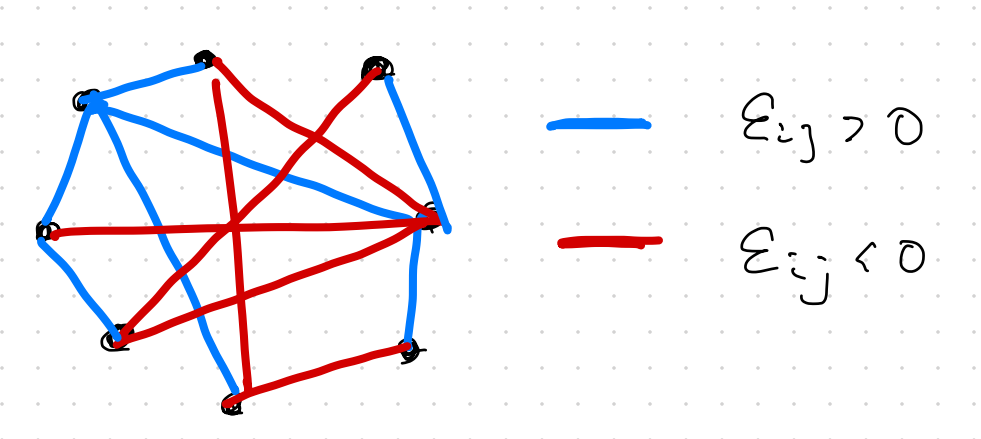
http://www.kniaz.net/software/RosNM.aspx

量子アニーリング

- 離散最適化問題

$$H = -J \sum_{i<j} \epsilon_{ij} \sigma_i^z \sigma_j^z$$

の基底状態配位と基底状態エネルギーを求めたい



量子アニーリング

- 横磁場を導入

$$H = -J \sum_{i<j} \epsilon_{ij} \sigma_i^z \sigma_j^z - \Gamma \sum_i \sigma_i^x$$

- 古典極限 $(J = 1, \Gamma = 0)$
  - 求めたい基底状態
- 量子極限 $(J = 0, \Gamma = 1)$
  - 2^N 個の全ての状態の重ね合わせ
- 量子アニーリング
  - $J + \Gamma = 1$ を保ったままで、$\Gamma = 1$ から $\Gamma = 0$ まで「ゆっくり」と減少させながら時間発展させる

$$J = 1 - \Gamma = t/T \qquad (0 \le t \le T)$$

  - $T \to \infty$ の極限で確率 1 で基底状態に収束

確率過程を用いた最適化

- 最急降下法 (steepest decent)
  - 初期状態をランダムに定める
  - 配位を少しだけ変化させる
  - エネルギー (コスト関数) が小さくなるなら採択、大きくなるなら棄却 $\Rightarrow$ 絶対零度での Metropolis 法と等価
  - 状態が変化しなくなるまでくり返す
  - 問題点：エネルギー極小状態にすぐに捕まってしまう
- 徐冷法 (simulated annealing)
  - いきなり温度を零にするのではなく少しずつ下げていく
  - どれくらいゆっくり下げれば良いか?

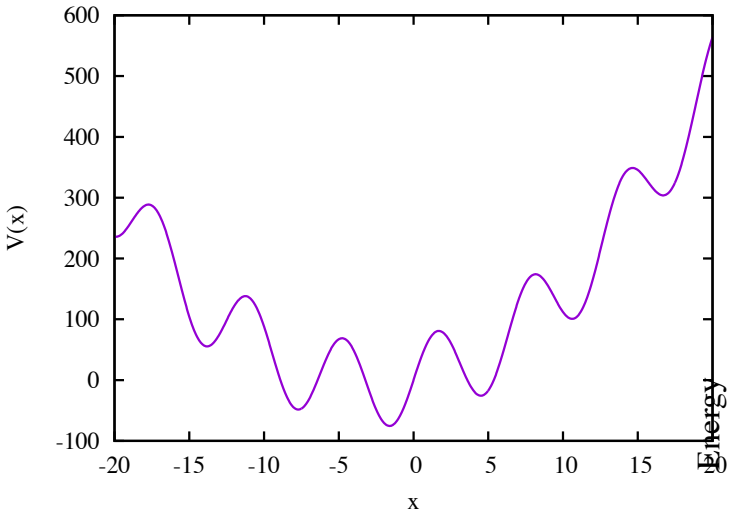
$$T(t) \geq cN/\log(t+2)$$

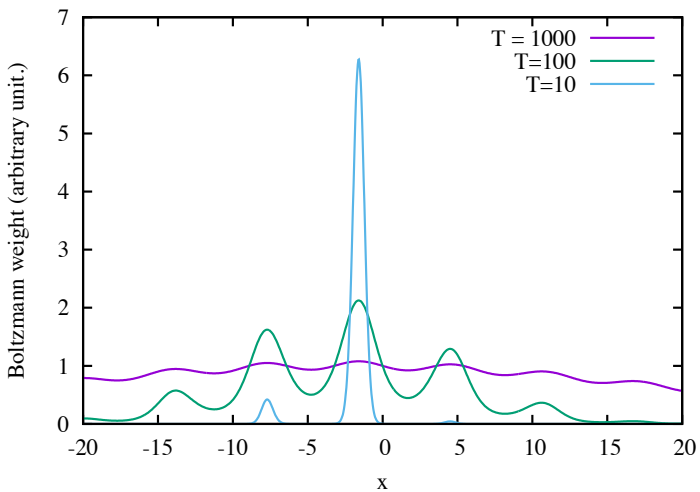
  - 実際には適当なスケジューリングで温度を下げ、何回か繰り返して最も良い結果を採択

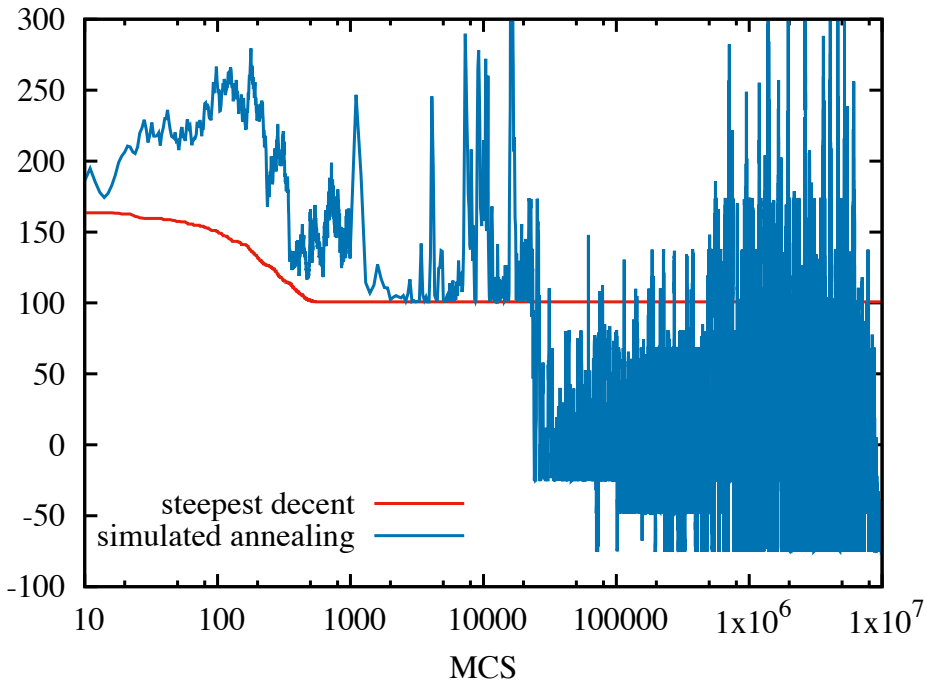
Metropolis 法

- 現在の配位 x から、試行配位 (trial configuration) x' を $x - \Delta \sim x + \Delta$ の一様分布より選ぶ
- 確率 $\min\left(1, \frac{e^{-\beta V(x')}}{e^{-\beta V(x)}}\right)$ で x' を採択 (accept)。棄却 (reject) された場合にはもとの x のまま
- 物理量の測定 (reject された場合にもカウントする)
- 採択確率 (acceptance probability) は、$\frac{e^{-\beta V(x')}}{e^{-\beta V(x)}+e^{-\beta V(x')}}$ でもよい
- 例: harmonic.c

最急降下法とシミュレーテッドアニーリング

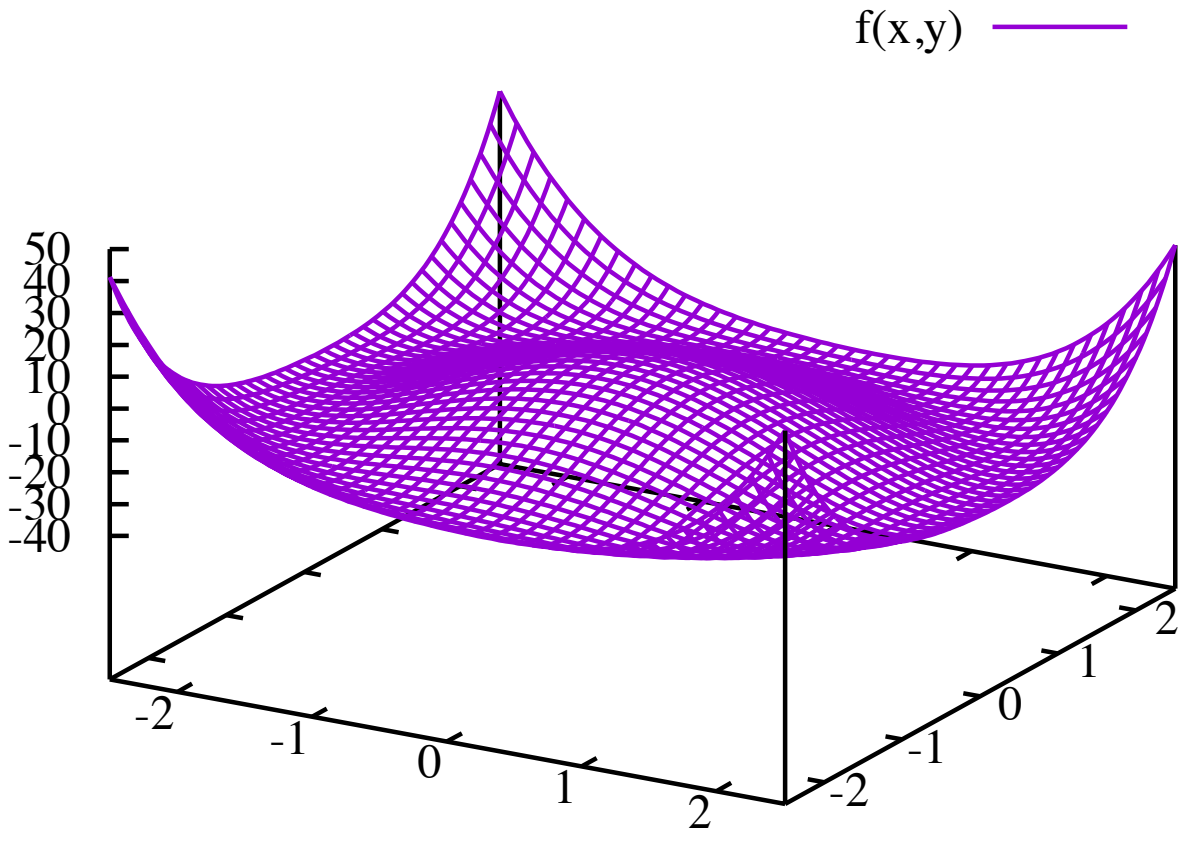




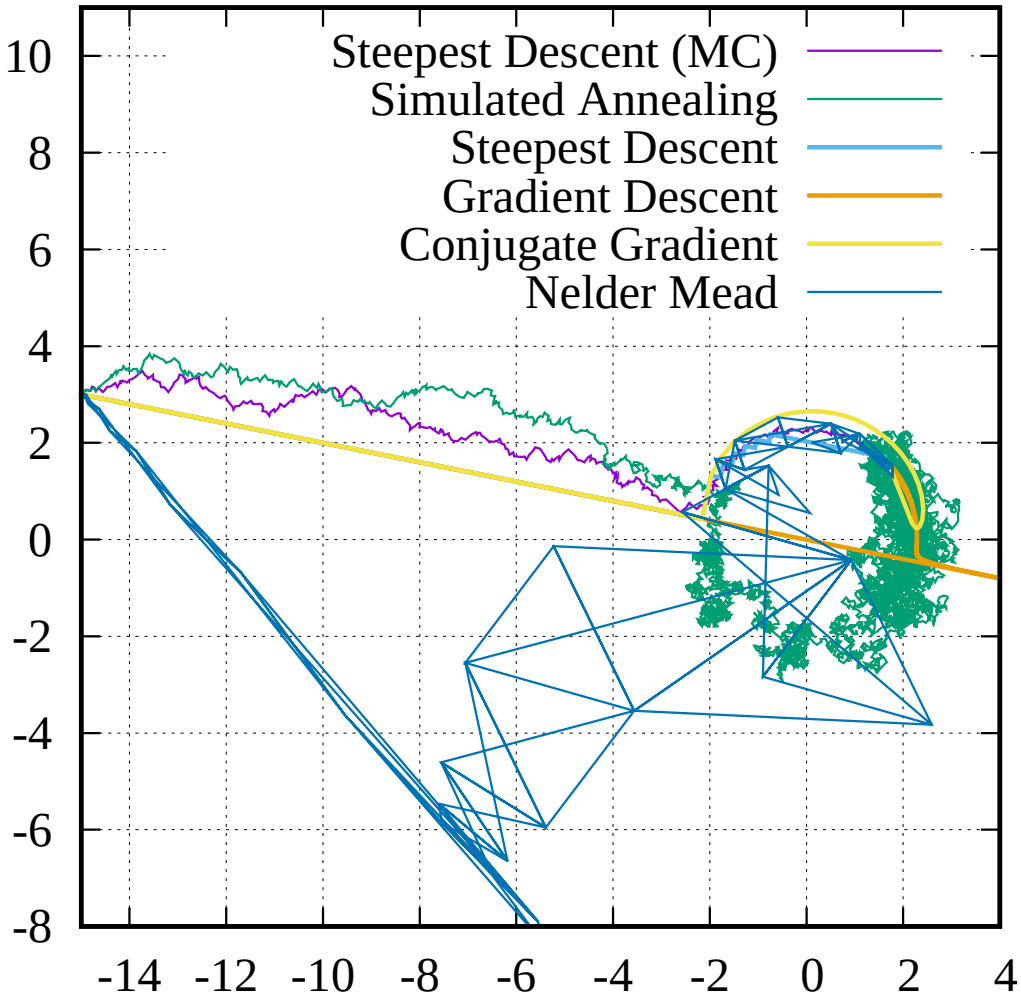


$$T(t) = 100 - \frac{99}{10^7}t$$

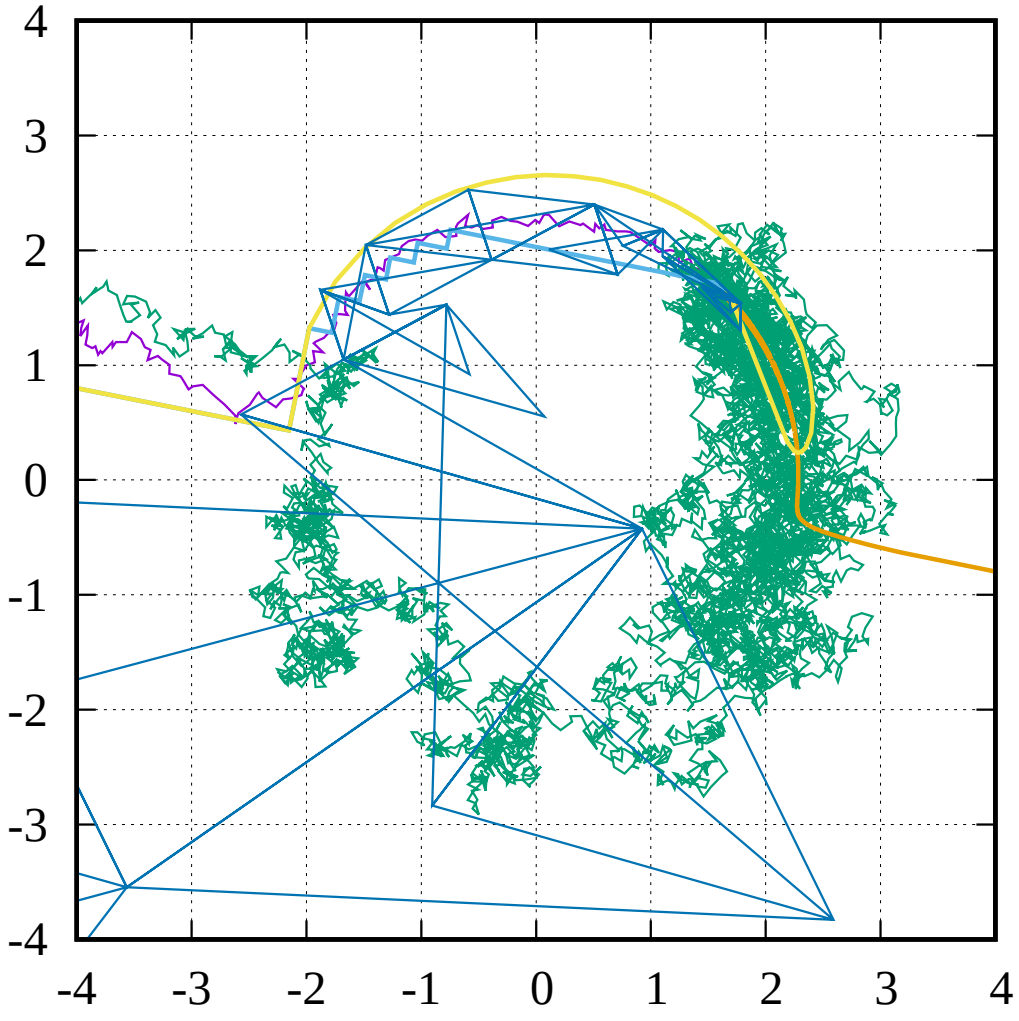
例題 (二次元の最適化)



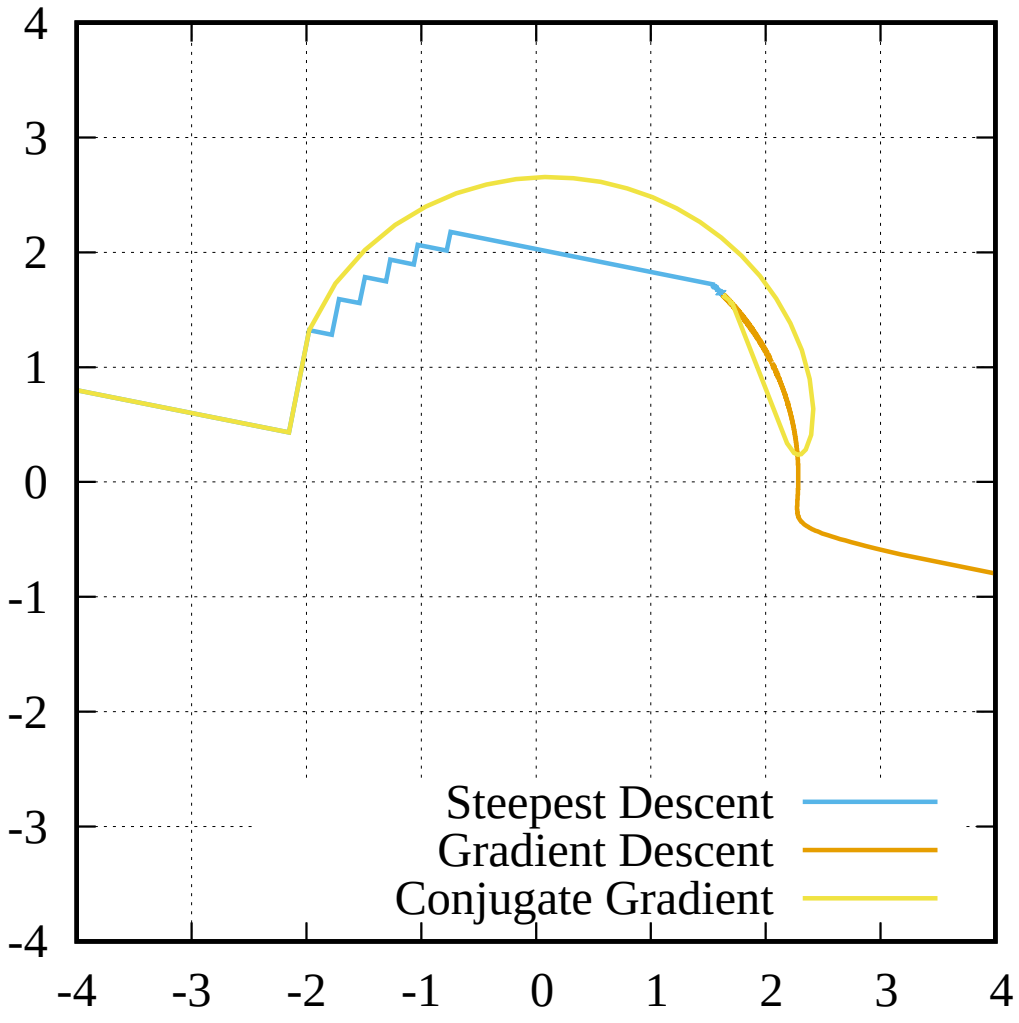
様々な最適化手法の比較 (1/4)



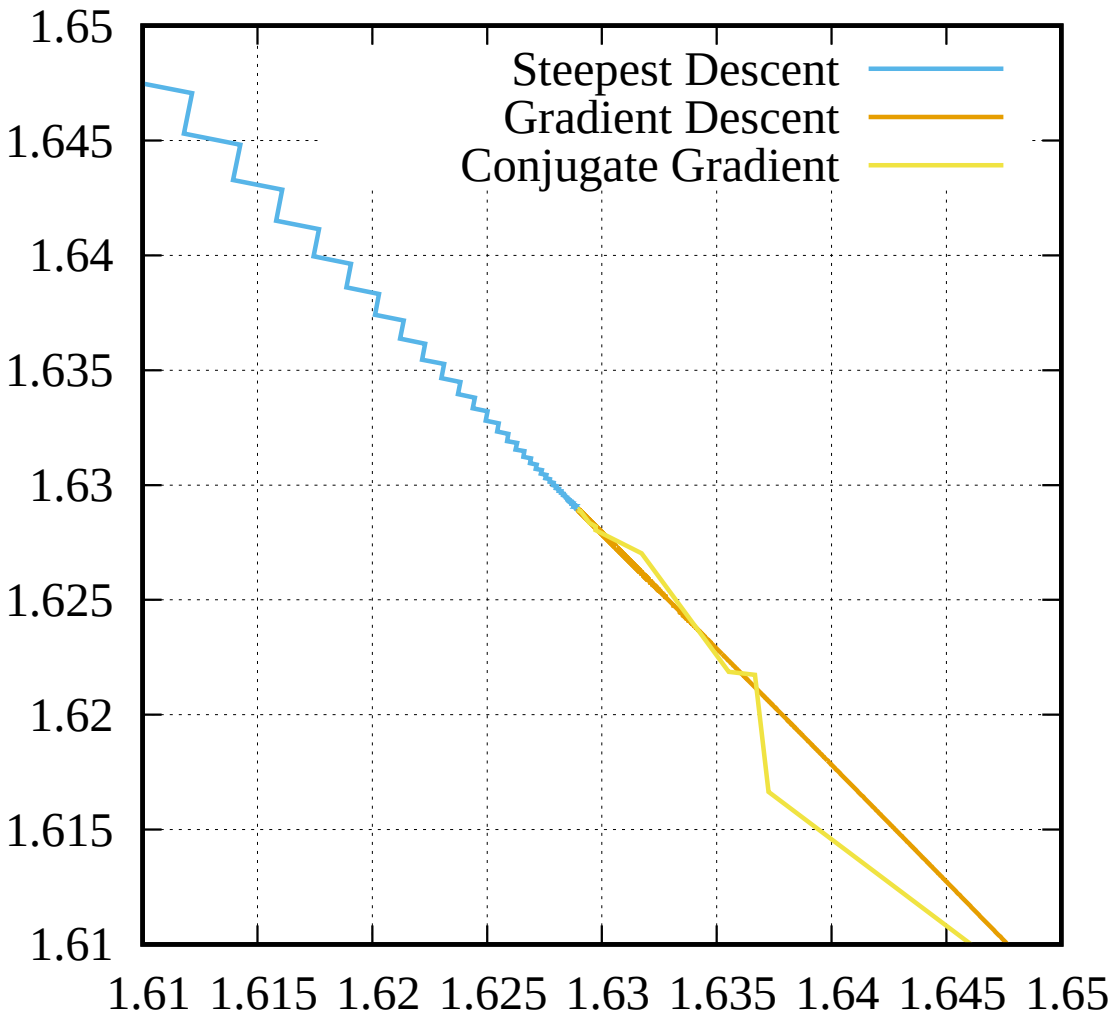
様々な最適化手法の比較 (2/4)



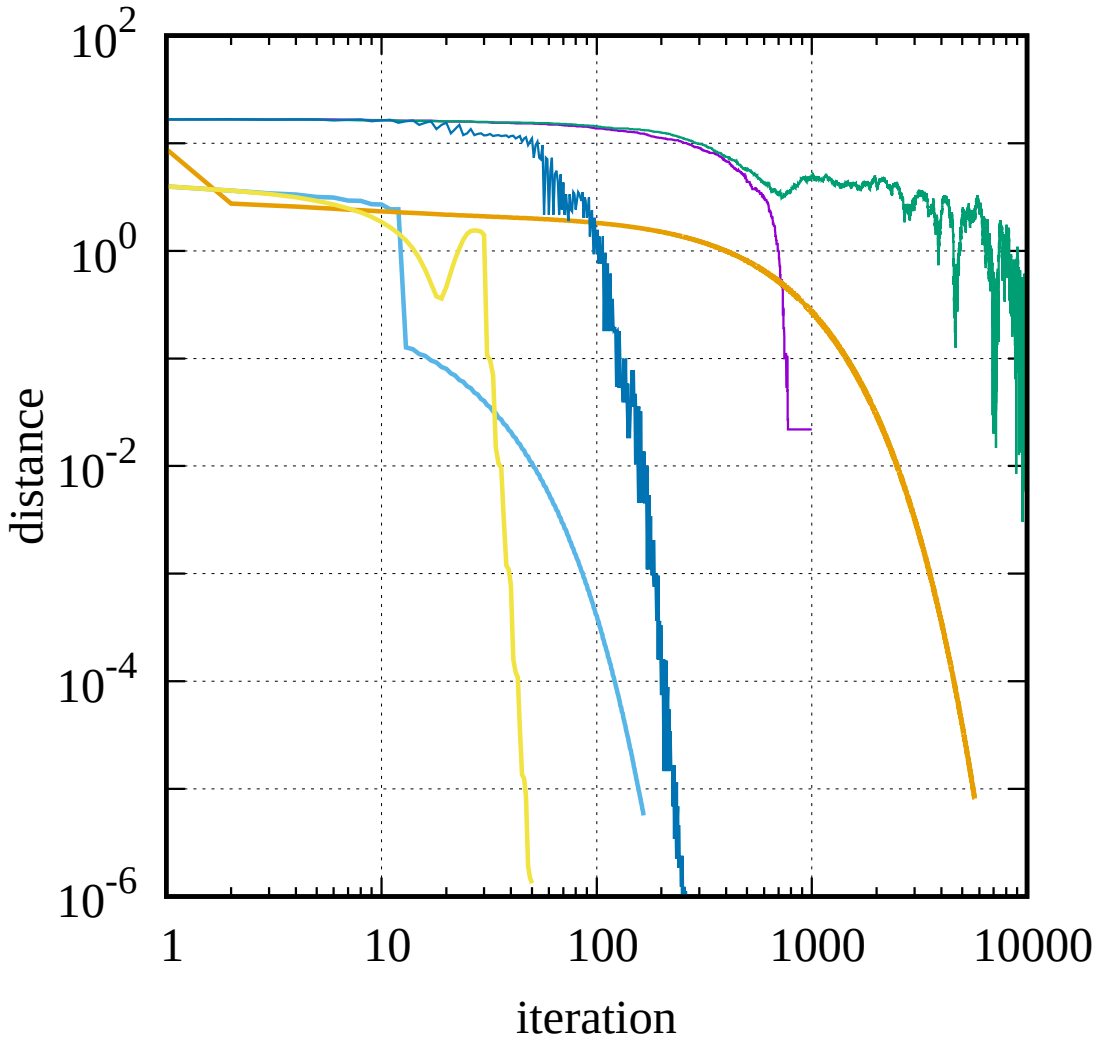
様々な最適化手法の比較 (3/4)



様々な最適化手法の比較 (4/4)



真の解への近づき方



計算機環境

- 教育用計算機システム
- 知の物理学クラスタ (ai)
  - 卒業まで利用可 (希望すれば大学院でも)
  - バッチシステムを使えば、かなり大規模な計算も可能
- 計算機利用・シミュレーションに関する質問は今後も歓迎

  `computer@phys.s.u-tokyo.ac.jp`