Brief Introduction to Numerical Optimization and its application to Molecular Conformation

Dongli Deng
Bedřich Sousedík (mentor)

Supported in part by NSF award DMS-1521563

April 30, 2018

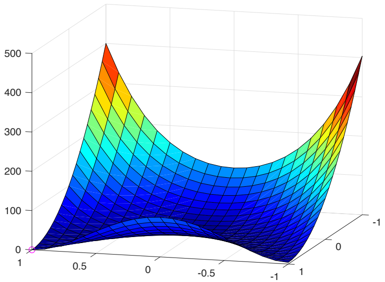
Introduction

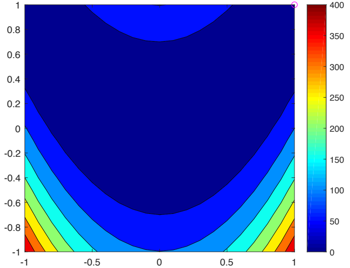
- We studied four methods:
  - Steepest Descend
  - Conjugate gradient
  - Newton's method
  - A quasi-Newton method
- We implemented the methods in MATLAB, and tested them on several small problems.
- We applied the methods to minimize Lennard-Jones potential.

Test function 1: (Ex. 2.1, Nocedal: Numerical Optimization, 2006)

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2 \qquad \text{(minimum at } (1, 1))$$

$$\nabla f = \begin{bmatrix} -400xy + 400x^3 - 2 + 2x \\ 200y - 200x^2 \end{bmatrix}, \qquad H = \begin{bmatrix} -400y + 1200x^2 + 2 & -400x \\ -400x & 200 \end{bmatrix}$$

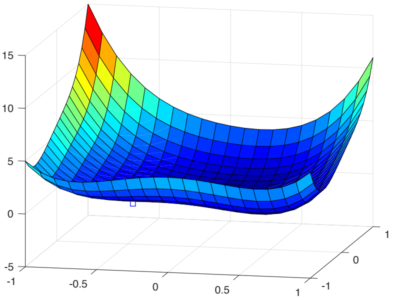


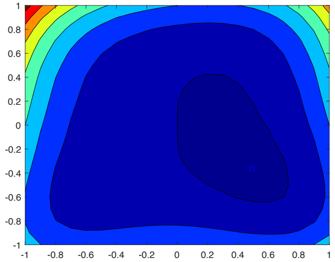


Test function 2: (Example 13.3, Sauer: Numerical analysis, Pearson, 2006)

$$f(x,y) = 5x^4 + 4x^2 - xy^3 + 4y^4 - x \qquad \text{(minimum at } (0.4923, -0.3643))$$

$$\nabla f = \begin{bmatrix} 20x^3 + 8xy - y^3 - 1 \\ 4x^2 - 3y^2x + 16y^3 \end{bmatrix}, \qquad H = \begin{bmatrix} 60x^2 + 8y & -3y^2 \\ -3y^2 & 48y^2 - 6yx \end{bmatrix}$$





Steepest Descend

$$x_{k+1} = x_k - sv$$

$s \dots$ step length (we can use Successive Parabolic Interpolation),
$v \dots$ direction of the steepest descend at x_k (given by $\nabla f(x_k)$)

Algorithm of Steepest Descend

for $n = 0, 1, 2, \ldots$ **do**
    v=$\nabla f(x_i)$
    Minimize $f(x - sv)$ for scalar $s = s^*$
    $x_{i+1} = x_i + s^* v$
end for

Steepest Descend for test problem 2 (Sauer)

Step	x	y	$f(x, y)$
1	1.0000	-1.0000	5.0000
5	0.1867	0.2136	-0.1444
10	0.3327	0.0418	-0.2530
15	0.4228	-0.2142	-0.4036
20	0.4877	-0.3561	-0.4573
25	0.4922	-0.3641	-0.4575
30	0.4923	-0.3643	-0.4575

Conjugate Gradient Method

Consider minimizing, starting with quadratic function

$$f(x) = \frac{1}{2}x^T Ax - x^T b$$

$$\nabla f = Ax - b$$

Finding the minimum is equivalent to solving $Ax = b$.

One-dimensional line search: given search direction d, find step length α so that the function $f(x + \alpha d)$ is minimized

$$0 = \nabla f \cdot d = (\alpha Ad - r)^T \cdot d$$

$$\alpha = \frac{r^T d}{d^T Ad} = \frac{r^T r}{d^T Ad}$$

$r \ldots$ residual of the linear system (given by $-\nabla f(x) = b - Ax$).

$$0 = \nabla f \cdot d = (A(x + \alpha d) - b) \cdot d$$

$$0 = \nabla f \cdot d = (Ax + \alpha Ad - b) \cdot d$$

Let

$$r = b - Ax$$

$$0 = (\alpha Ad - r)^T \cdot d$$

$$\alpha d^T Ad - r^T d = 0$$

$$\alpha = \frac{r^T d}{d^T Ad}$$

Algorithm of Conjugate Gradient Method

Let x_0 be the initial guess and set $d_0 = r_0 = -\nabla f$.
for $n = 0, 1, 2, \ldots$ **do**
    $\alpha_i = \alpha$ that minimizes $f(x_{i-1} + \alpha d_{i-1})$
    $x_i = x_{i-1} + \alpha_i d_{i-1}$
    $r_i = -\nabla f(x_i)$
    $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
    $d_i = r_i + \beta_i d_{i-1}$
end for

Conjugate Gradients for test problem 2 (Sauer)

Step	x	y	$f(x, y)$
1	0.0998	0.4114	0.0247
2	0.5372	-0.3240	-0.4324
3	0.5093	-0.3812	-0.4557
4	0.4944	-0.3776	-0.4569
5	0.4900	-0.3644	-0.4575
...			
10	0.4923	-0.3643	-0.4575

Newton's Method in one Dimension

For solving $f(x) = 0$ we use

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

Applying the same idea to $f'(x) = 0$ gives

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}.$$

This can be equivalently written as

$$f''(x_i)\,(x_{i+1} - x_i) = -f'(x_i).$$

Newton's Method in higher dimension

Given an initial guess x_0, for $k = 0, 1, \ldots,$ solve

$$H(x_k)v = -\nabla f(x_k),$$

update

$$x_{k+1} = x_k + v.$$

Here

$$\nabla f = \left[\frac{\partial f}{\partial x_1}(x_1, ..., x_n), \quad \ldots \quad \frac{\partial f}{\partial x_n}(x_1, ..., x_n) \right]^T,$$

$$H_f = \begin{bmatrix} \frac{\partial^2 f(x_1,...,x_n)}{\partial x_1^2} & \ldots & \frac{\partial^2 f(x_1,...,x_n)}{\partial x_1 \partial x_n} \\ .. & & \\ .. & & \\ .. & & \\ \frac{\partial^2 f(x_1,...,x_n)}{\partial x_1 \partial x_n} & \ldots & \frac{\partial^2 f(x_1,...,x_n)}{\partial x_n^2} \end{bmatrix}.$$

Quasi Newton's method

$H \ldots$ from now on, an approximation to the inverse of the Hessian,

Update x as

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

$$\begin{aligned} \alpha_k & \quad \ldots \quad \text{step length (from backtracking line search/Wolfe condition)}, \\ p_k & = -H_k \nabla f_k \qquad \text{(update of the search direction)}, \\ s_k & = x_{k+1} - x_k \qquad \text{(change of the displacement)}, \\ y_k & = \nabla f_{k+1} - \nabla f_k \qquad \text{(change of gradients of the functions)} \end{aligned}$$

Update the (inverse of) the Hessian as

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

where

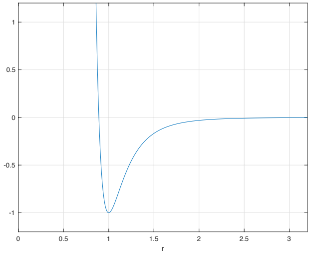
$$\rho_k = \frac{1}{(y_k^T s_k)}.$$

Algorithm of BFGS

Given starting point x_0, convergence tolerance $\epsilon > 0$
choose (inverse) Hessian approximation H_0 (commonly I)
k $\leftarrow$ 0
while $\mid \nabla f_k \mid > \epsilon$; **do**
    $p_k = -H_k \nabla f_k$
    $x_{k+1} = x_k + \alpha_k p_k$
    Compute H_{k+1} (updating H_k)
    k $\leftarrow$ k+1
end while

Lennard-Jones Potential

$$U(r) = \frac{1}{r^{12}} - \frac{2}{r^6}, \qquad r \ldots \text{ distance between two atoms}$$



General form with n atoms

$$U(x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}} \right),$$

$r_{ij} \ldots$ distance between atoms i and j

Gradient of the Lennard-Jones potential

$$U = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} U_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}} \right)$$

and, for example,

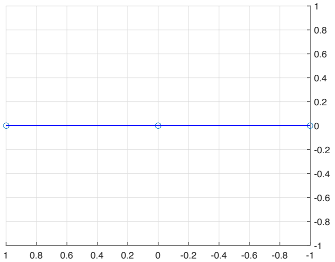
$$\frac{\partial U_{ij}}{\partial x_i} = \frac{-12(x_i - x_j)}{r_{ij}^{14}} + \frac{12(x_i - x_j)}{r_{ij}^{8}}$$

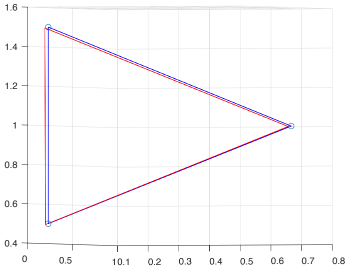
where

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

Three atoms

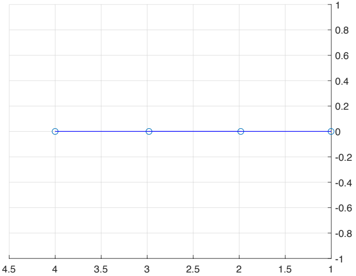
Left: initial guess $(0,-5,0)(0,0,0)(0,5,0)$ (not global minimum)
Right: initial guess $(0,0,0)(0,0,2)(1,1,1)$,
comparison of FMINUNC and our implementation of BFGS

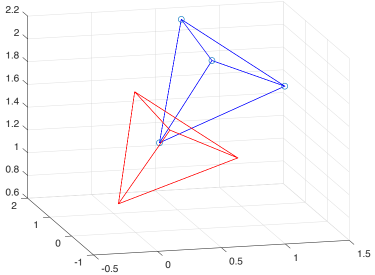




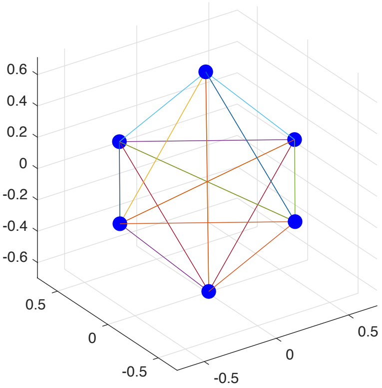
Four atoms

Left: initial guess $(0,-5,0)(0,0,0)(0,5,0)(0,10,0)$ (not global min.)
Right: initial guess $(0,0,0)(0,0,2)(1,1,1)(2,3,4)$,
comparison of FMINUNC and our implementation of BFGS





Six atoms



(We got the same result using FMINUNC in MATLAB.)