CSE/ECE 848 Introduction to Evolutionary Computation

Module 1, Lecture 2, Part 2

Existing Point-based Optimization Methods

Kalyanmoy Deb, ECE

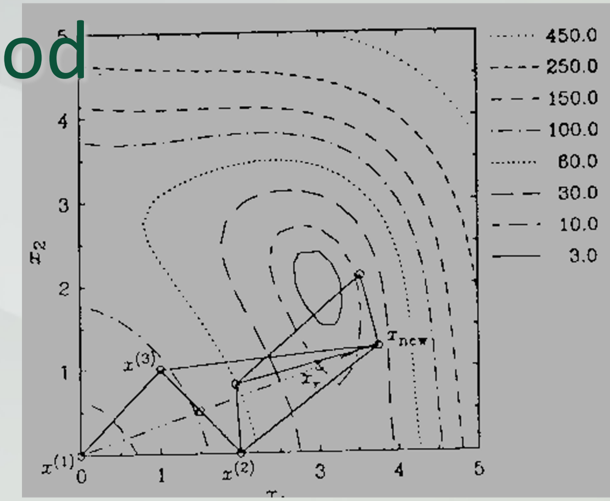
Koenig Endowed Chair Professor

Recall Module 1, Lecture 2, Part 1

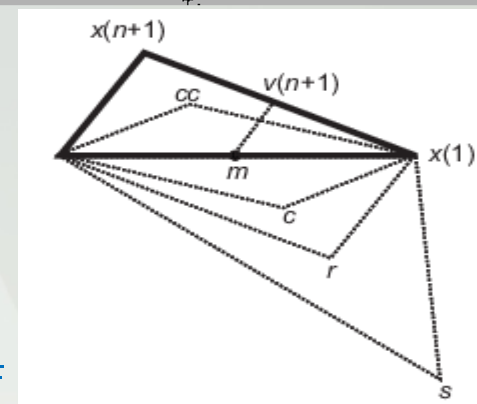
- Point-based Unconstrained Optimization:
  - Several line-searches to reach the optimum
- Each line-search is a single-variable search
- Bound and locate the minimum along a line
- 1. Bounding phase, followed with Golden Section
- 2. Hybrid Golden Section and Parabolic search

- Part 2: Multi-variable Unconstrained optimization
  - Direct Search: Use of objective function $f(\boldsymbol{x})$ only
  - Gradient Search: Use of first and/or Second-derivative with $f(\boldsymbol{x})$

Nelder-Mead's Simplex Search Method



- $x^{(i)}$ denote $n+1$ points
- Sort them in ascending of $f\left(x^{(i)}\right)$
- Compute the reflected point
  - $r = m + (m - x^{(n+1)}), m$ is avg of first n pts
- If $f\left(x^{(1)}\right) \le f(r) \le f\left(x^{(n)}\right)$, select r (reflect)
- If $f(r) < f\left(x^{(1)}\right)$, $s = m + 2\left(m - x^{(n+1)}\right)$
  - If $f(s) < f(r)$, select s (expand), else select r (reflect)
- If $f(r) \ge f\left(x^{(n)}\right)$
  - If $f(r) < f\left(x^{(n+1)}\right)$, $c = m + (r - m)/2$
    - If $f(c) < f(r)$ select c (contract outside), else Go To Step F
  - If $f(r) \ge f(x^{(n+1)})$, $cc = m + (x^{(n+1)} - m)/2$; If $f(cc) < f(x^{(n+1)})$, select cc (contract inside), else Go To Step F
- F: $v(i) = x^{(1)} + (x^{(i)} - x^{(1)})/2, i = 2, \dots, n+1$, simplex is with $x^{(1)}$ (shrink)



Matlab: fminsearch()

Nonlinear programming solver. Searches for the minimum of a problem specified by

$$\min_x f(x)$$

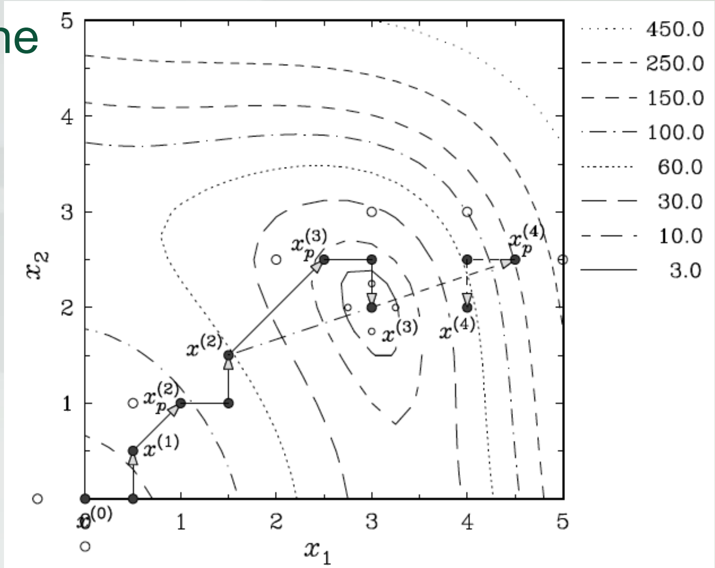
[x = fminsearch(fun,x0,options)]

Reference: Lagarias, J. C., J. A. Reeds, M. H. Wright, and P. E. Wright. 1998. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal of Optimization*, *9*(1), 112–147.

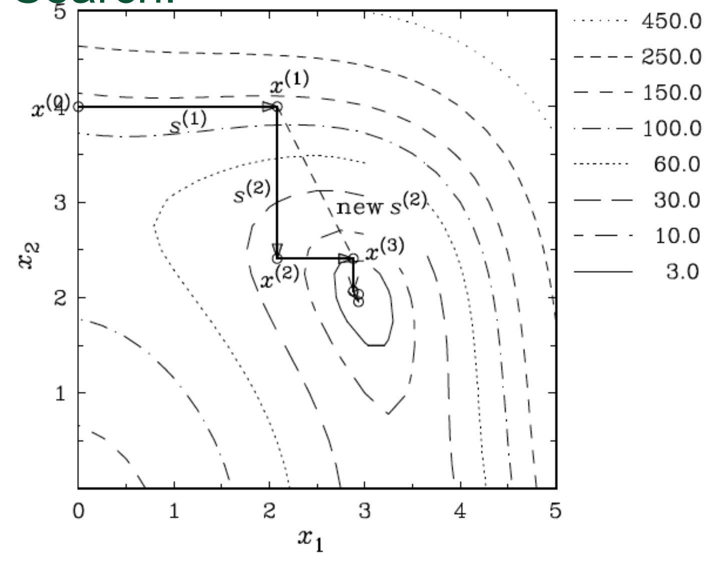
Other Direct Search Methods

- Hooke-Jeeves Pattern Search method
- Conjugate direction method
- Randomized methods (Luus-Jaakola)
- Hill-climbing methods
- Trust region methods
  - Search restricted in a neighborhood of current best solution
- Mostly local and dependent on initial point(s)

Does not use Line Search:

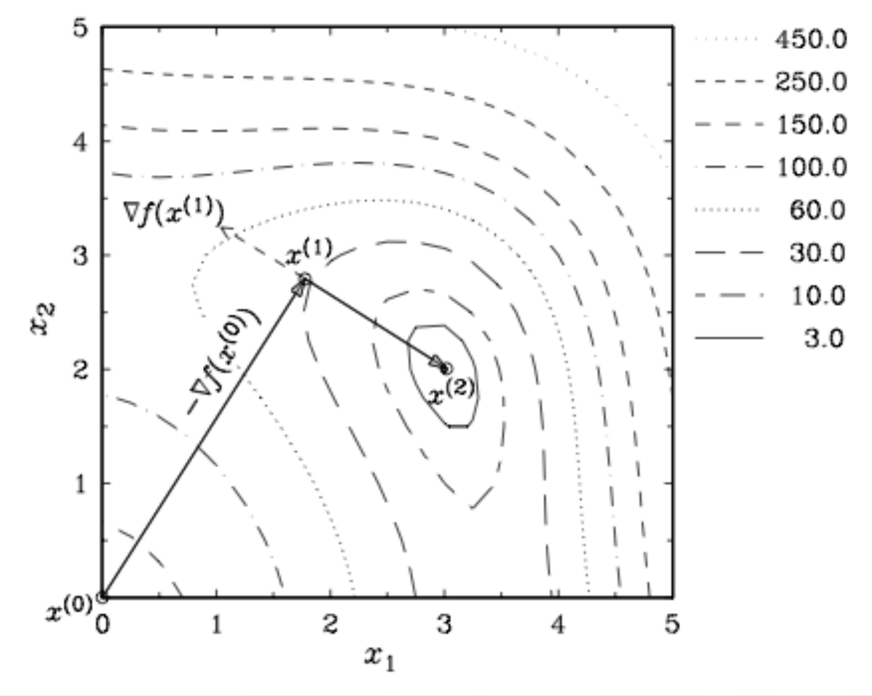


Uses Line Search:



Gradient-Based Methods

- Gradient vector takes *2n* solution evaluations
- Hessian matrix takes *2n² +1* evaluations
- Cauchy's Steepest-descent method
  - $\mathbf{s}^{(t)} = -\nabla f(\mathbf{x}^{(t)})$
  - Reduces the function value maximally



References:

1. Deb, K. (1995). *Optimization methods for engineering design: Algorithms and Examples*. Delhi: Prentice-Hall. (Available from Amazon).

2. Reklaitis, Gintaras V., A. Ravindran, and Kenneth M. Ragsdell. 1983. *Engineering optimization: Methods and applications*. New York: Wiley.

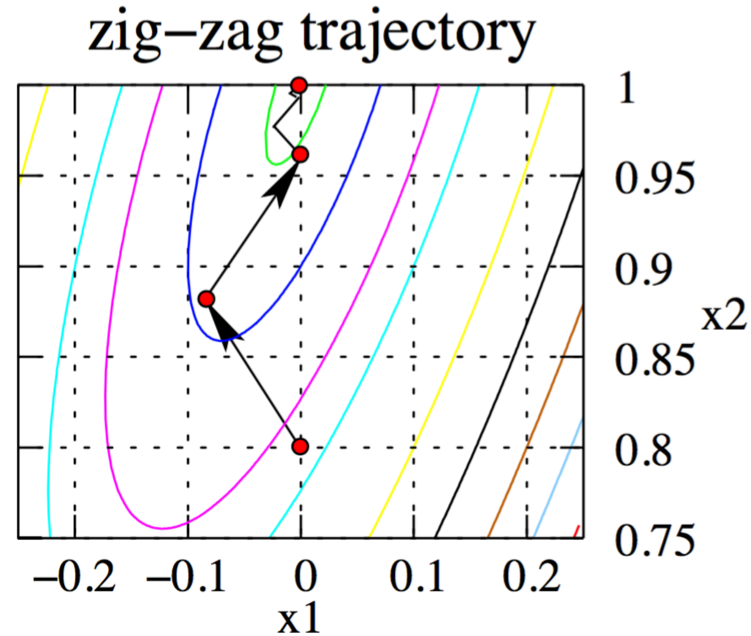
Example Problem (Steepest-Descent Method)

2. The steepest descent method is applied to the following function which is to be minimized:

$$f(x_1, x_2) = 2x_1^2 - 2x_1x_2 + x_2^2 + 2x_1 - 2x_2.$$

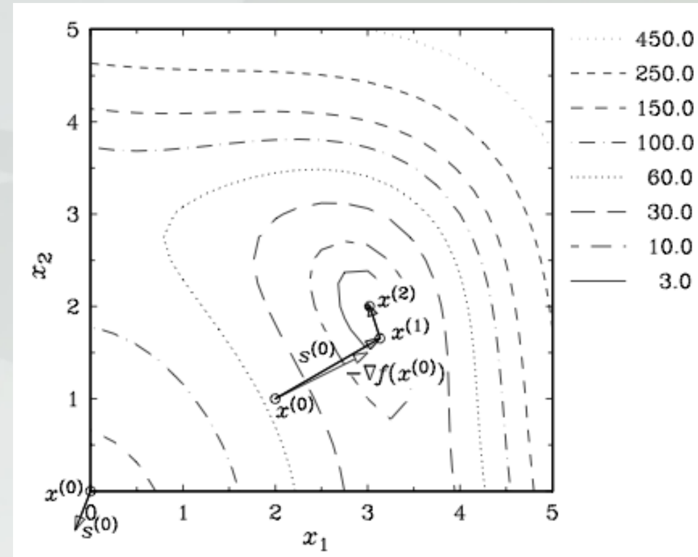
Start with $\mathbf{x}^{(0)} = (0, (1 - 1/5))^T$. Show that after two iterations of the steepest descent search, the point $\mathbf{x}^{(2)} = (0, (1 - 1/5^2))^T$ is obtained.

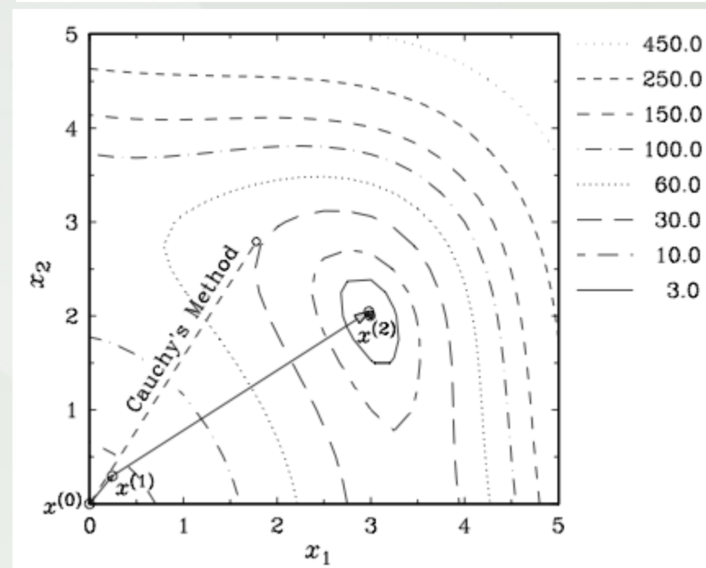
Two consecutive directions are always orthogonal to each other



Newton’s and Marquardt’s Methods

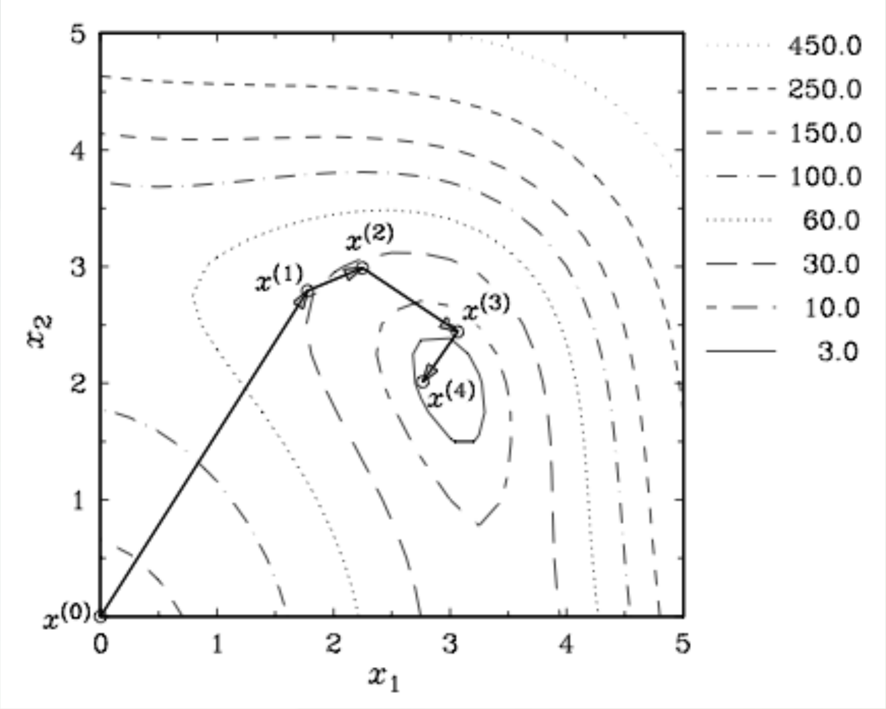
- Newton’s method require Hessian:
  - $\mathbf{s}^{(t)} = -\nabla^2 f(\mathbf{x}^{(t)})^{-1}\nabla f(\mathbf{x}^{(t)})$
  - Computationally expensive
- Marquardt’s method is a compromise
  - Start with Cauchy’s and end with Newton’s method





Quasi-Newton Methods

- Newton's method is expensive
- Q-Newton's method uses gradient vectors to compute Hessian at opt.
  - $A^0 = I, A^1 \rightarrow A^2 \rightarrow \ \dots \ \rightarrow$ Hessian($\mathbf{x}^*$)
- DFP, BFGS methods exist
- Convergence proof for quadratic problems exists



DFP:

$$A^{(k)} = A^{(k-1)} + \frac{\Delta x^{(k-1)} \Delta x^{(k-1)T}}{\Delta x^{(k-1)T} \Delta e(x^{(k-1)})} - \frac{A^{(k-1)} \Delta e(x^{(k-1)}) (A^{(k-1)} \Delta e(x^{(k-1)}))^T}{\Delta e(x^{(k-1)})^T A^{(k-1)} \Delta e(x^{(k-1)})}$$

$$\Delta x^{(k-1)} = x^{(k)} - x^{(k-1)}$$

$$\Delta e(x^{(k-1)}) = e(x^{(k)}) - e(x^{(k-1)})$$

End of Module 1, Lecture 2, Part 2

- Multi-variable, point-based optimization methods
- Direct and gradient-based methods
  - Most methods use a line search iteratively
- Part 3:
  - Point-based Constrained Optimization