559

Non-linear Minimization

Fish 559; Lecture 8

Introduction

- Non-linear minimization (or optimization) is the numerical technique that is used by far most frequently in fisheries assessments.
- The problem:
  - Find the vector $\underline{\theta}$ so that the function $f(\underline{\theta})$ is minimized (note: maximizing $f(\underline{\theta})$ is the same as minimizing $-f(\underline{\theta})$).
  - We may place bounds on the values for some of the elements of $\underline{\theta}$ (e.g. some must be positive).

Minimizing a Function-I

- By definition, for a minimum:

$$\left.\frac{\partial f(\underline{\theta})}{\partial \underline{\theta}}\right|_{\underline{\theta}_{opt}} = \underline{0}$$

- This problem statement is deceptively simple. There is *no* perfect algorithm. The art of non-linear minimization is to know which method to use and how to determine whether your chosen method has converged to the correct solution. (Also phasing, starting values, etc)

Minimizing a Function-II

- There are many techniques to find the minimum (maximum) of a function depending on:
  - the cost of evaluating the function;
  - the cost of programming;
  - the cost of storing intermediate results;
  - whether analytical or numerical derivatives are available;
  - whether bounds are placed on some of the parameters.

Analytic Approaches-I

- Sometimes it is possible to solve the differential equation directly. For example:

$$SSQ = \sum (y_i - \hat{y}_i)^2; \qquad \hat{y}_i = a + b\,x_i$$
$$= \sum (y_i - a - b\,x_i)^2$$

- Now:

$$\frac{dSSQ}{da} = -2\sum (y_i - a - bx_i) = 0 = \sum y_i - aN - b\sum x_i$$
$$\frac{dSSQ}{db} = -2\sum (y_i - a - bx_i)x_i = 0 = \sum x_i\, y_i - a\sum x_i - b\sum (x_i)^2$$
$$\Rightarrow b = \frac{N\sum x_i\, y_i - \sum x_i \sum y_i}{N\sum (x_i)^2 - \sum x_i \sum x_i}; \quad a = \frac{\sum y_i - b\sum x_i}{N}$$

Analytical Approaches-II

- Use analytical approaches whenever possible. Finding analytical solutions for some of the parameters of a complicated model can substantially speed up the process of minimizing the function.
- For example: *q* for the Dynamic Schaefer model:

$$SSQ = \sum (\ell n I_t - \ell n \hat{I}_t)^2; \qquad \hat{I}_t = q(\hat{B}_t + \hat{B}_{t+1})/2 = q\tilde{B}_t$$

$$\frac{dSSQ}{dq} = -\sum (\ell n I_t - \ell n[q\tilde{B}_t]) \frac{2}{q} = 0$$

$$\Rightarrow \sum \ell n(I_t / \tilde{B}_t) = N \, \ell n q$$

$$\Rightarrow q = \exp\{\tfrac{1}{N} \sum \ell n(I_t / \tilde{B}_t)\}$$

Analytic Approaches-III

- The "analytical approach" runs into two main problems:
  - The differentiation is very complicated for typical fisheries models.
  - The resultant equations may not have an analytical solution, e.g:

$$y_i = a e^{-b x_i} + \varepsilon_i \qquad\qquad \varepsilon_i \sim N(0;\sigma^2)$$

- We need to find a way to minimize a function numerically.

Newton's Method - I (Single variable version)

- We wish to find the value of x such that $f(x)$ is at a minimum.

1. Guess a value for x
2. Determine whether increasing or decreasing x will lead to a lower value for $f(x)$ (based on the derivative).
3. Assess the slope and its change (first and second derivatives of f) to determine how far to move from the current value of x.
4. Change x based on step 3.
5. Repeat steps 2-4 until no further progress is made.

Newton's Method - II (Single variable version)

- Formally:

1. Set $x = x_0$

2. Compute $f'(x) = \frac{df(x)}{dx}$; $f''(x) = \frac{d^2 f(x)}{dx^2} = \frac{df'(x)}{dx}$

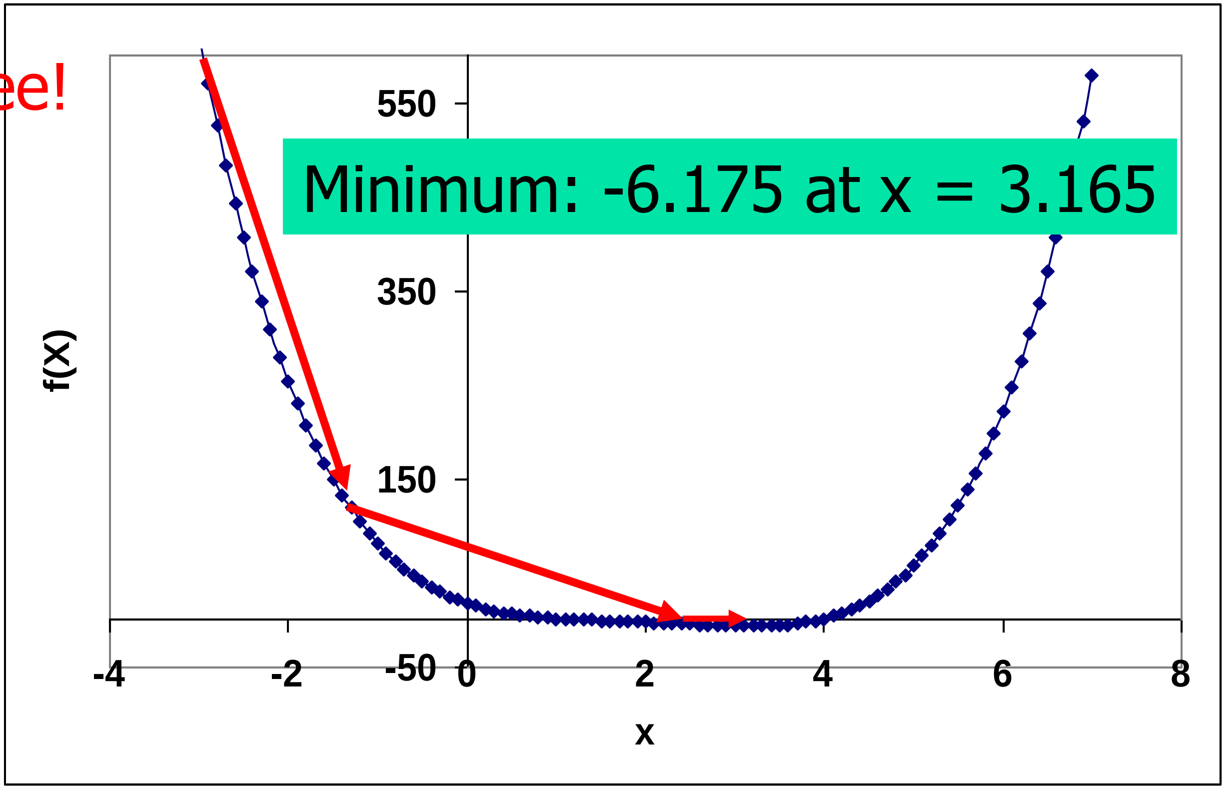
3. Modify x to $x \to x - \frac{f'(x)}{f''(x)}$

4. Repeat steps 2 and 3 until x stops changing

- Note: Newton's method may diverge rather than converge!

Minimize: 2+(x-2)^4-x^2

This is actually quite a nasty function – differentiate it and see!



Convergence took 17 steps in this case.

Multidimensional methods

- We will focus on multidimensional methods because most fisheries problems are (highly) multidimensional.
- Derivative free:
  - Nelder-Mead (Simplex); and (built into optim())
  - Direction-set methods (Powell's method).
- Derivatives required:
  - Conjugate gradient methods (Fletcher's method); and
  - Quasi-Newton methods.
- There is usually no reason not to use derivative-based methods if you can compute derivatives (cheaply).
- The focus of this lecture will be on the derivative-free methods.

The Simplex Method-I

- Slow and robust, this method “crawls” (amoeba-like) towards the solution. It requires no derivatives and can deal with bounded parameters.

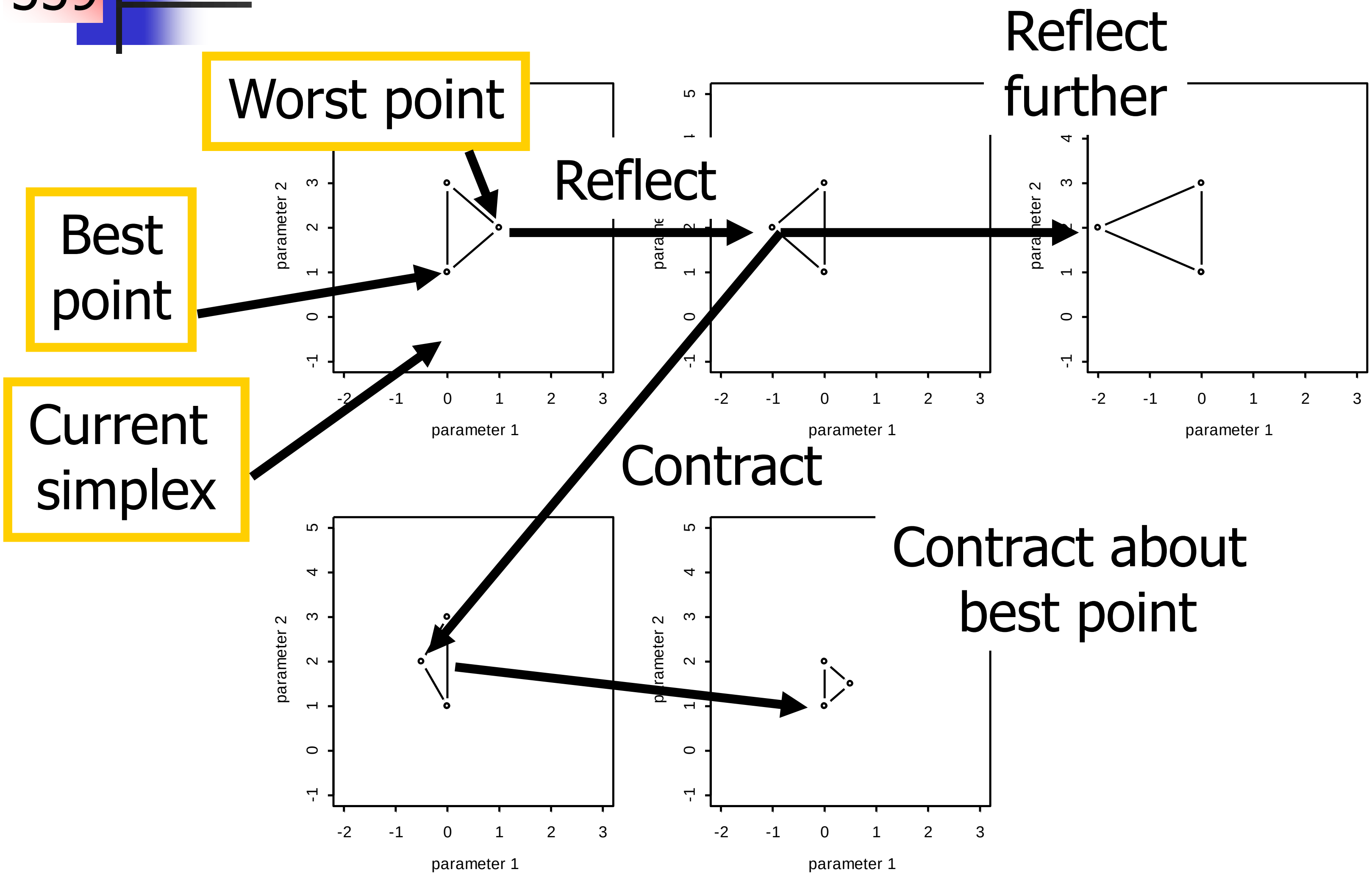
The Simplex Method-II (an overview)

1. Set up a "simplex" (a set of *N*+1 points, where *N* is the dimension of the parameter vector) and evaluate *f* at each vertex (two dimensions is a triangle).
2. Find the point that has the highest value of *f* (the worst point) and examine a point reflected away from this point.
3. If the new point is better than the best point then try again in the same direction.

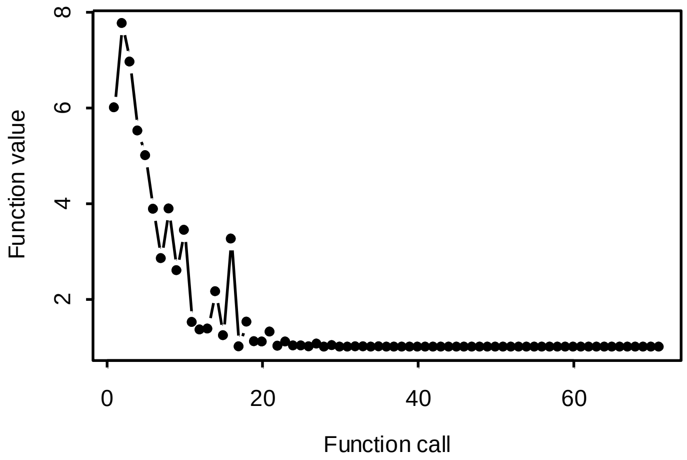
The Simplex Method-III (an overview)

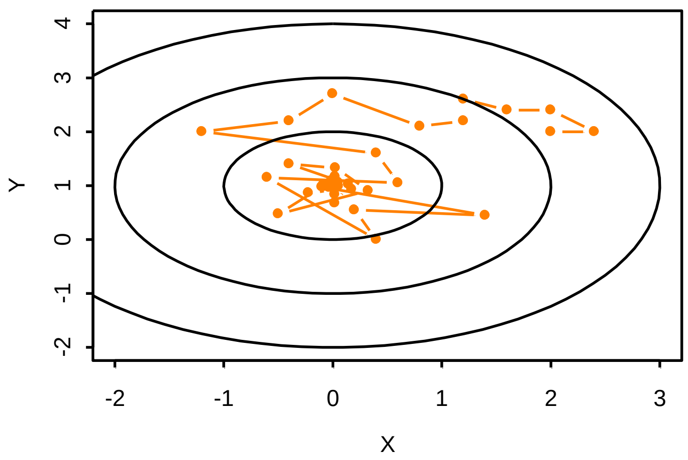
4. If the reflected point is worse than the second worst point:
    1. replace the worst point if the reflected point is better than the worst point;
    2. contract back away from the reflected point;
    3. if the contracted point is better than the highest point replace the highest point; and
    4. if the contracted point is worse that the worst point contract towards the best point.
5. Replace the worst point by the reflected point.
6. Repeat steps 2-5 until the algorithm converges.

The Simplex Method-IV (N=2)



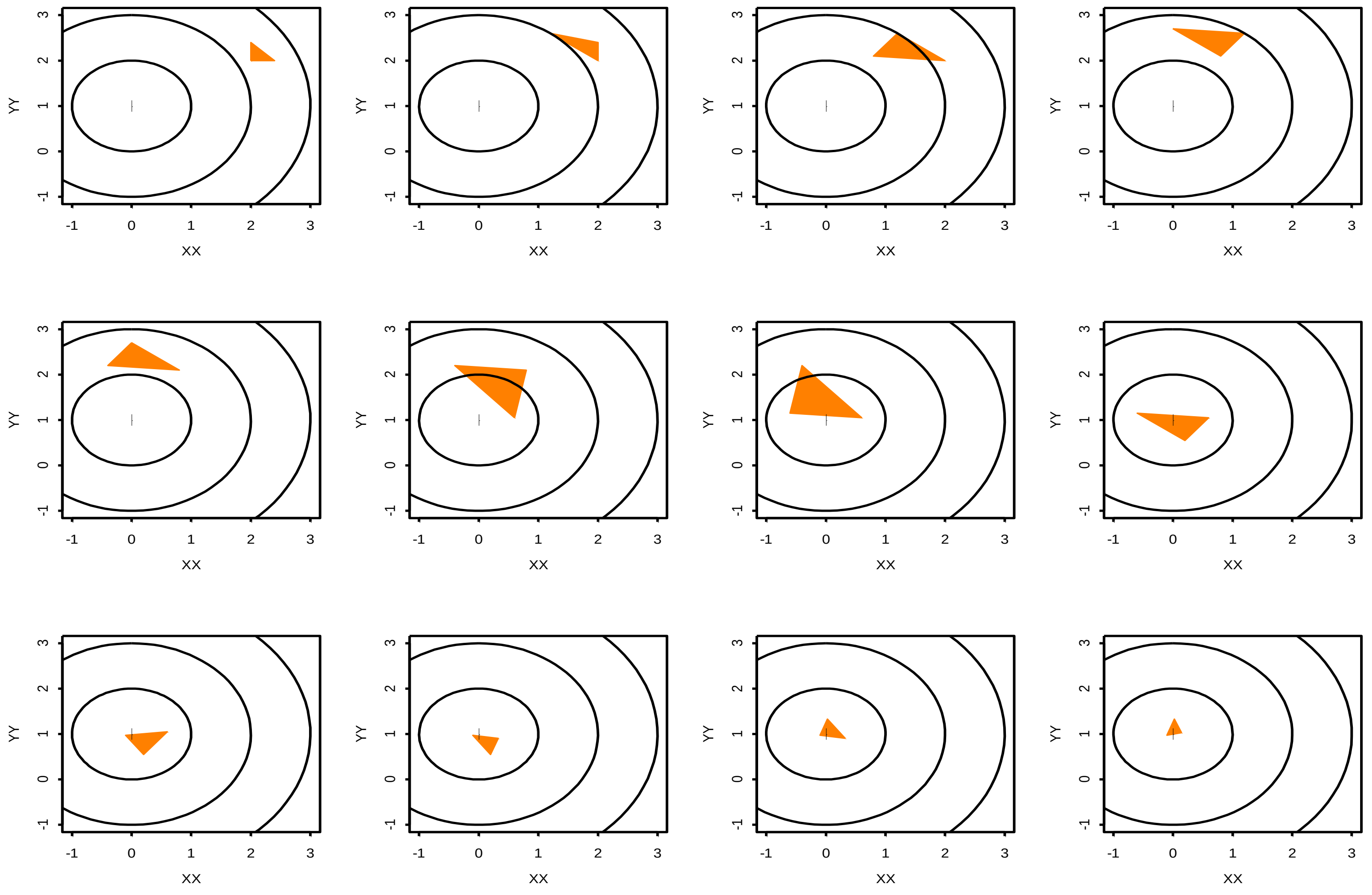
An Example of the Simplex Method-I





Minimize: $f(x,y) = 1 + x^2 + (1-y)^2$

An Example of the Simplex Method-II



Direction Sets (The Basic Idea-I)

1. Select a starting point, P_0 (in N dimensions) and set i=0.
2. Minimize f in each of N search directions, u_j (e.g. the unit vectors – this is a single dimensional search). The function value at the end of each search is $P_{i,j.}$. Note: $P_{i+1}=P_{i,N}$.
3. Set $u_{j-1}=u_j$ and $u_N=P_{i+1}-P_i$. (changes the orthogonality of the search vectors)
4. Repeat steps 2-3 until the algorithm converges.

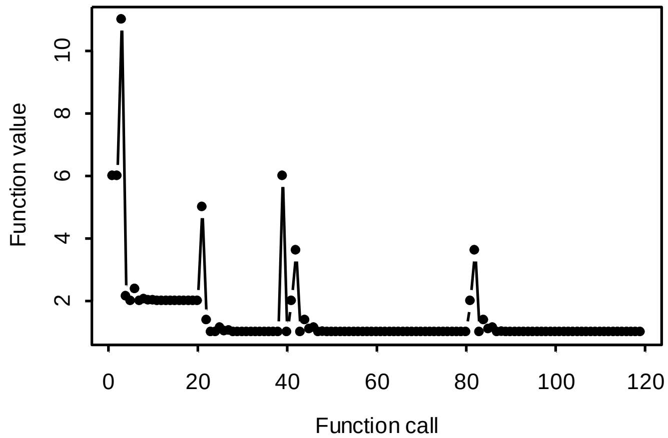
Direction Sets (The Basic Idea-II)

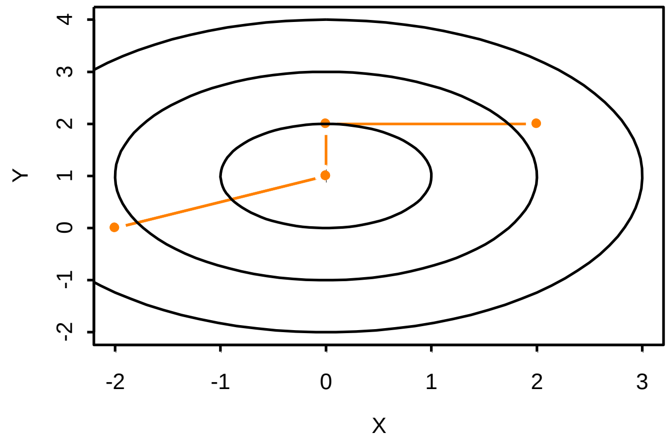
- Advantages:
  - Replaces a multi-dimensional search by a set of single-dimensional searches.
- Disadvantages:
  - Without step 3, the algorithm is very slow.
  - With step 3, the algorithm can fail to converge.
- These problems are solved by Powell’s method.

Powell's Method (Steepness decent)

1. Select a starting point, P_0 (in N dimensions) and set i=0.
2. Minimize f in each of N search directions. The function value at the end of each search is $P_{i,j}$. Note: $P_{i+1}=P_{i,N}$.
3. Set $P_e=(2P_{i,N}-P_i)$ and compute $f(P_e)$.
4. If $f(P_e) > f(P_i)$ then replace the search direction of maximum change to $P_{i+1}-P_i$.
5. Repeat steps 2-4 until the algorithm converges.

An Example of the Powell's Method





Minimize: $f(x,y) = 1 + x^2 + (1-y)^2$

Problem with this method is that it can only move in one direction. Does well for quadratics, but not banana-shaped surfaces

Using Derivatives (Steepest Descent Methods)

1. Select a starting point, P_0 (in N dimensions) and set i=0.
2. Compute the local downhill gradient, $-\nabla f(P_i)$,at P_i, (vector of derivatives)
3. Minimize f from P_i along this direction.
4. Repeat steps 2-3 until the algorithm converges.

- This method can, however, be very slow for problems with long narrow valleys.
- This method combines line minimization and the use of derivatives.
- Non-derivative methods have to work out where function is changing the most, derivatives have this information already

Using Derivatives

(Conjugate gradient methods, available in optim())

- At each step in the previous algorithm, we would prefer to try a direction that is *conjugate* to the previous direction, i.e. if the direction at step i g_i then $g_i \cdot g_{i+1} = 0$.
- We won't provide the formulae to achieve this (but see pages 302-5 of Numerical Recipes).
- To apply this type of algorithm you need the ability to compute the gradient and to perform line minimization.
- Conjugate gradient methods don't necessarily need derivatives.

Using Derivatives (Variable metric methods)

- The basic result that drives these methods is $P_{i+1} = P_i + (H_i)^{-1}(-\nabla f(P_i))$ where *H* is the Hessian matrix. In one dimension, this is just Newton's method that we saw earlier.
- Computing the Hessian matrix can be very demanding computationally so variable metric methods approximate it numerically.
- Hessian matrix with a single variable is the second derivative, making this a generalization of the Newton method to N dimensions
- BFGS derivative based method