Unconstrained Optimization: Line Search Algorithms

AE 6310: Optimization for the Design of Engineered Systems

Spring 2017

Dr. Glenn Lightsey

Lecture Notes Developed By Dr. Brian German

Georgia Institute of Technology

Daniel Guggenheim School of Aerospace Engineering

Line Search Algorithms

In the previous two sets of notes, we have examined

- Optimization along a given search direction
- Different types of search directions

We will now examine particular line search algorithms.

Each algorithm has a different approach to select a new search direction at each iteration.

Some algorithms require particular types of line search methods; others can be implemented with any line search method.

Types of Line Search Methods

Line search methods are categorized by the quality of the local approximation of the function that they use to determine $\boldsymbol{s_k}$:

Zeroth-order methods

- Use only values of the function $f(\boldsymbol{x})$ itself
- Some consider zeroth-order methods to be direct search methods

First-order methods

- Use values of $f(\boldsymbol{x})$ and $\nabla f(\boldsymbol{x})$
- May use approximations of $H(\boldsymbol{x})$ but are still of first-order accuracy

Second-order methods

- Use values of $f(\boldsymbol{x})$,$\nabla f(\boldsymbol{x})$, and $H(\boldsymbol{x})$

Methods that We Will Discuss

Zeroth-order methods

- Univariate Search
- Powell's Method

First-order methods

- Steepest Descent
- Fletcher-Reeves Conjugate Gradient
- Broyden, Fletcher, Goldfarb, Shanno (BFGS) Quasi-Newton Method

Second-order methods

- Newton's Method

There are many more algorithms! We have time to discuss just a few.

Univariate Search

Perhaps the simplest approach to selecting a search direction is to move in one of the coordinate variable directions.

If we systematically select a different coordinate variable direction in each iteration, then this method is called *univariate search.*

Univariate Search Algorithm

1. Select an initial point, $\boldsymbol{x_0}$

2. Select a coordinate variable direction (typically x_1)

3. Determine whether the $(+)$ or $(-)$direction along that coordinate line corresponds to a descent direction. This can be accomplished with a "probe step."

4. Find α^* along the coordinate line in the descent direction using a line search method that requires only f, not $df/d\alpha$.

Univariate Search Algorithm

5. Move to the $\boldsymbol{x_n}$ defined by α^* along the search direction.

6. Select the next coordinate variable direction; if we reach the end of the list, start over with x_1.

7. Repeat steps 3-6 until a convergence criterion has been met.

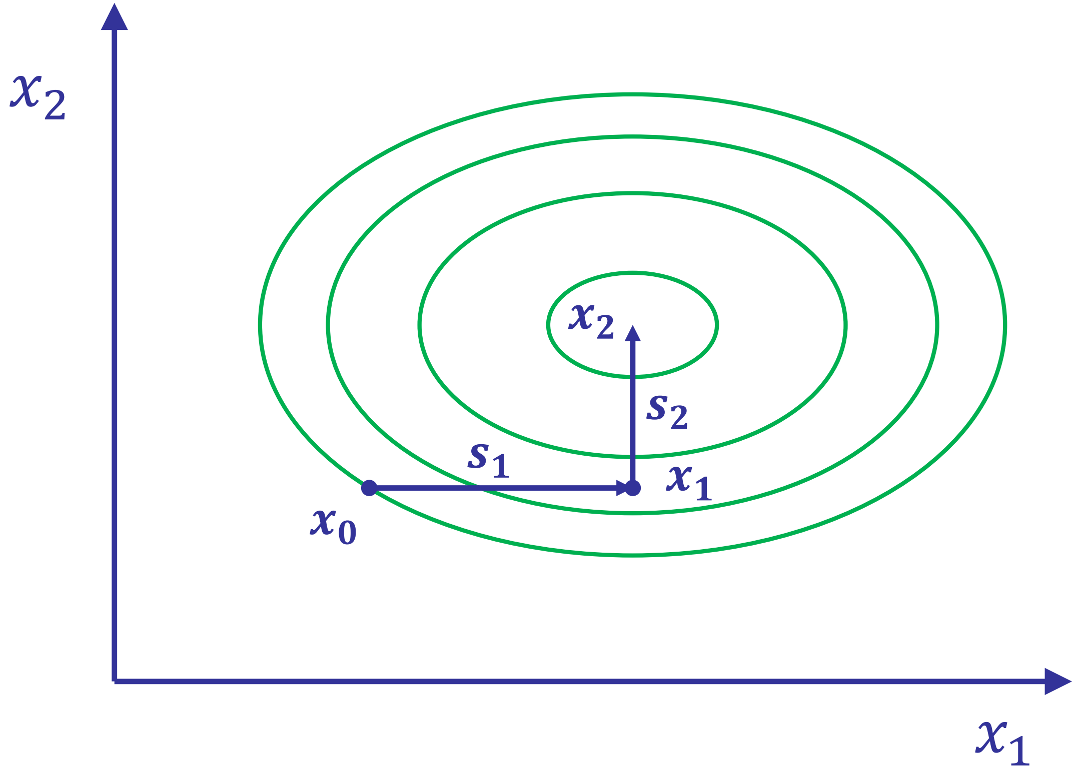
Univariate Search

An example progression of univariate search vectors for a 4-D problem is as follows:

$$\boldsymbol{s_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{s_2} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{s_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \boldsymbol{s_4} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

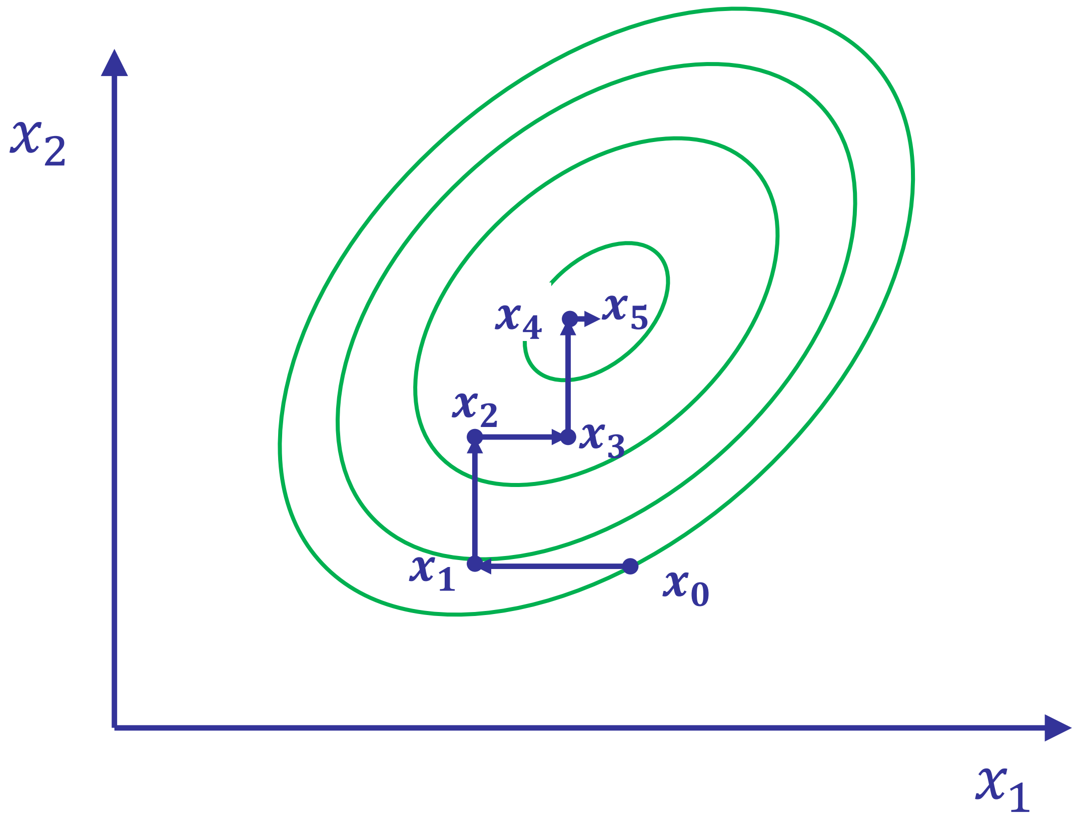
Univariate Search

Let's look at an example in which the line search method will converge rather well:

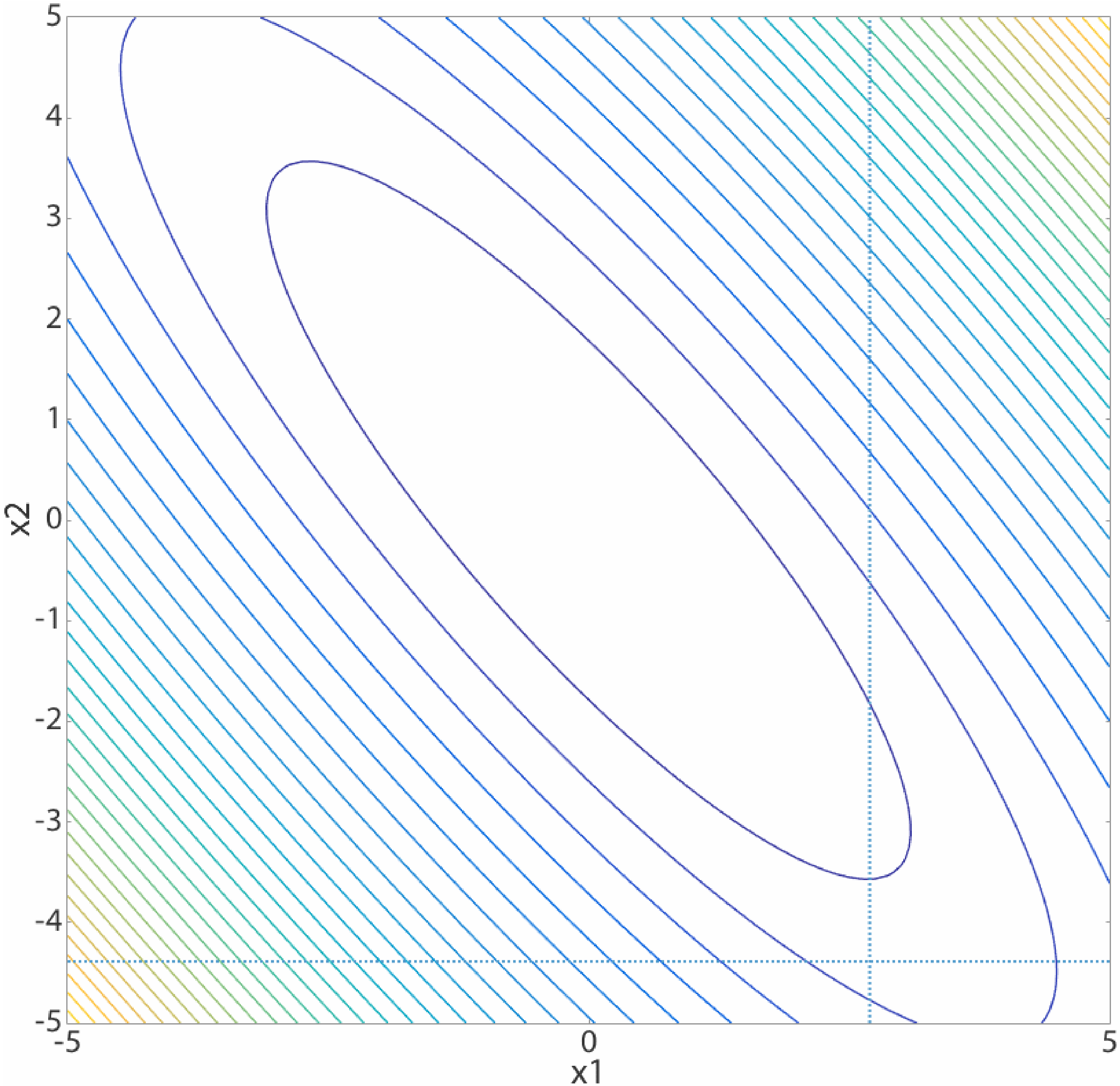


Univariate Search

But if the principal axes of the function contours were rotated from the coordinate axes, it would take more iterations:



Univariate Search



Variations on Univariate Search

An alternative form of the univariate search is to select the next coordinate variable direction as the one at the current point for which $\frac{\partial f}{\partial x_i}$ is greatest in magnitude.

The derivative information can be inferred by a “probe step” or from a gradient calculation.

If we use gradient information in the line search and the determination of the search direction, the univariate search becomes a first-order method.

Powell’s Method

Powell’s method is a modification of the univariate search based on *conjugate directions.*

The approach helps to “correct” the behavior of univariate search for cases in which the principal axes of the quadratic approximation of the function do not lie along a coordinate direction.

Powell's Method

1. Select initial point, $\boldsymbol{x_0}$
2. Begin with a set of n coordinate (univariate) vectors $\boldsymbol{s_i}$, $i = 1, \ldots, n$
3. Perform one line search along each $\boldsymbol{s_i}$ successively, moving to the minimum point α_i^* along the search direction each time.
4. Create a conjugate direction by summing the n vectors in the set involved in the iteration,

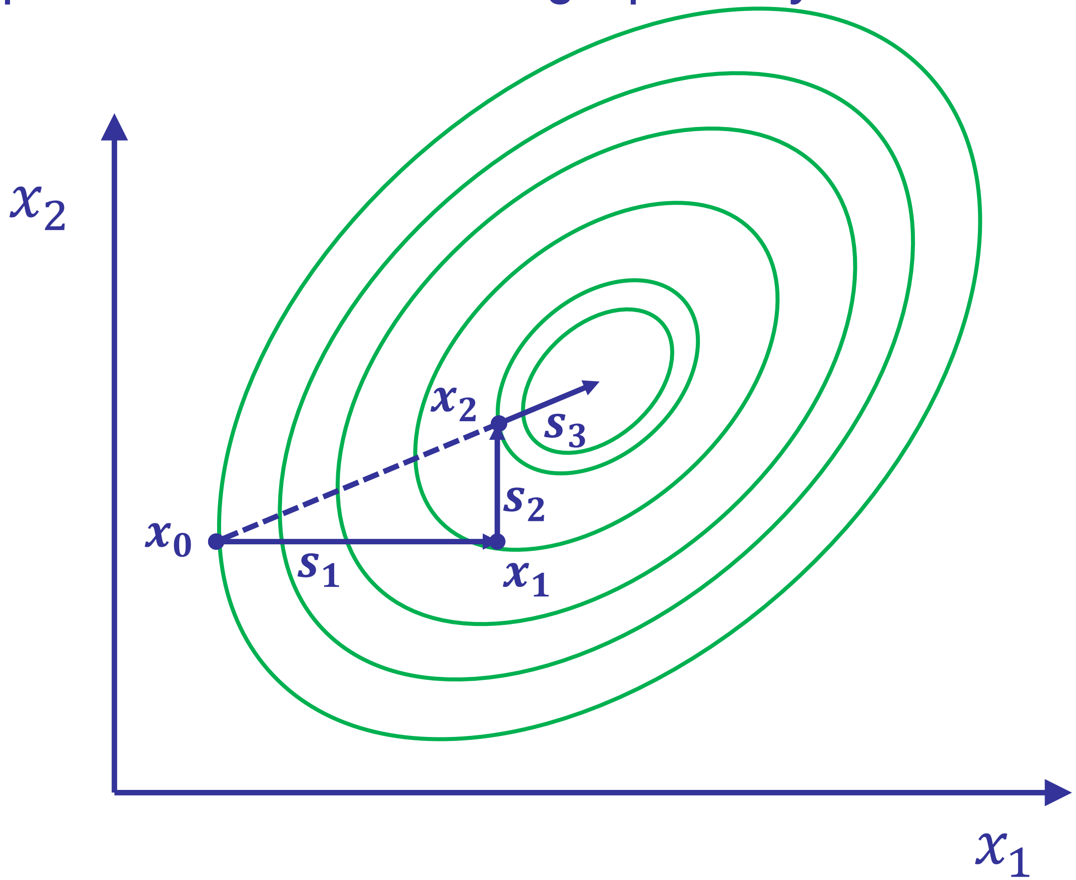
$$\boldsymbol{s_{n+1}} = \sum_{i=1}^{n} \alpha_i^* \boldsymbol{s_i}$$

Powell's Method

5. Search along the conjugate direction to determine α_{n+1}^{*} and move to the minimum point.
6. Update the set of vectors by discarding the 1st element, shifting other elements one to the left, and introducing $\boldsymbol{s_{n+1}}$ as the last element. This set of n+1 line searches is called an <u>iteration</u> of Powell's method.
7. Repeat steps 3 through 7 until a convergence criterion is met, resetting the set of vectors to the coordinate directions every n+1 iterations to prevent the search directions from becoming increasingly parallel.

Powell's Method

We can interpret Powell's method graphically as follows:



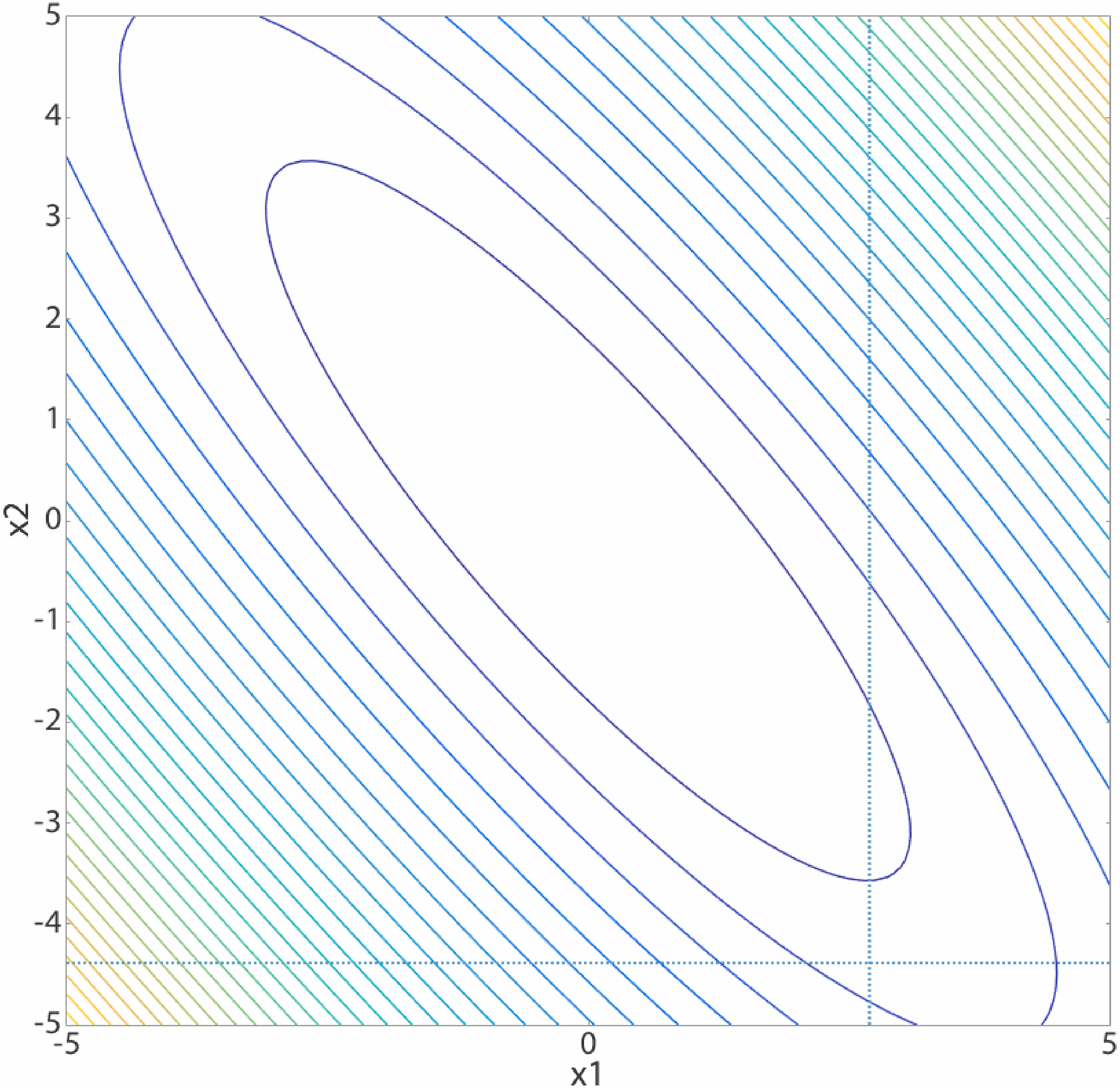
Search directions $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$ were the 2 coordinate directions and $\boldsymbol{s_3}$ was the first conjugate direction

Convergence of Powell’s Method

In general, for an n-D quadratic function with a minimum, Powell’s method will converge to the minimum after n conjugate directions have been formed, requiring n^2 line searches.

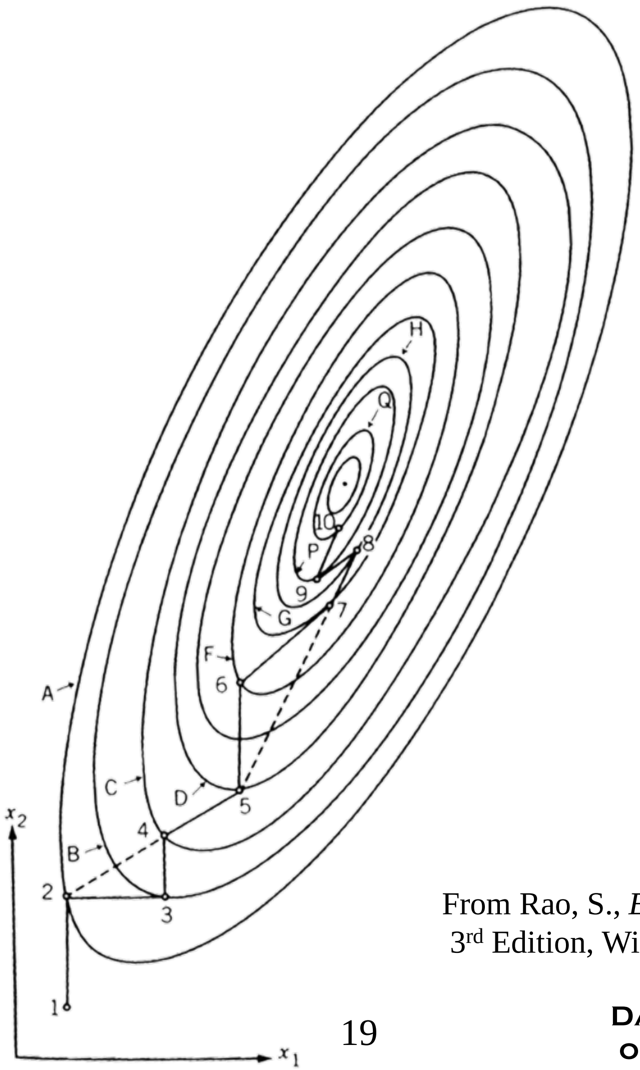
Powell’s method is therefore said to demonstrate *quadratic convergence.*

Powell's Method



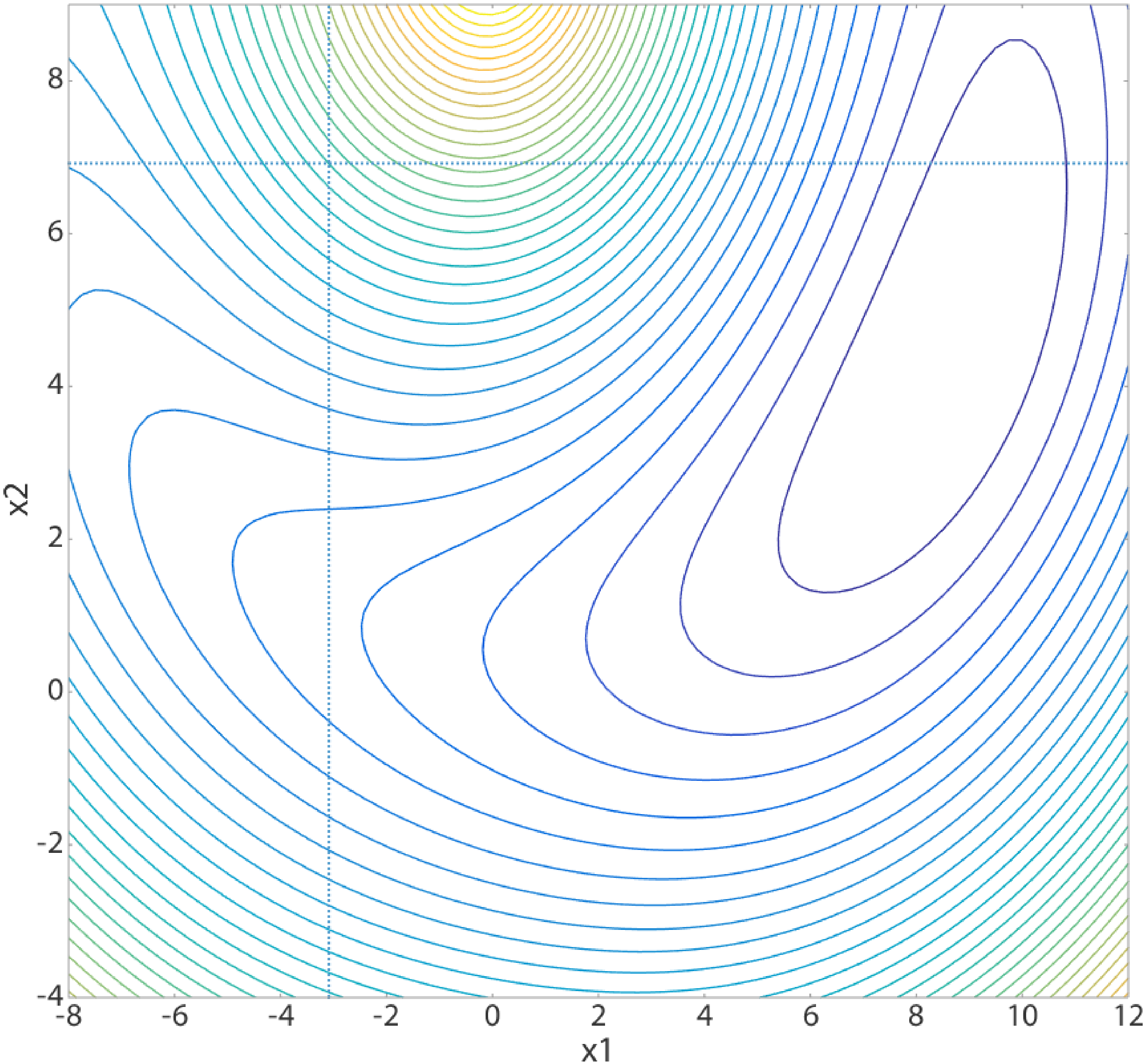
Powell's Method

For a more complicated (non-quadratic) function, convergence is slower:

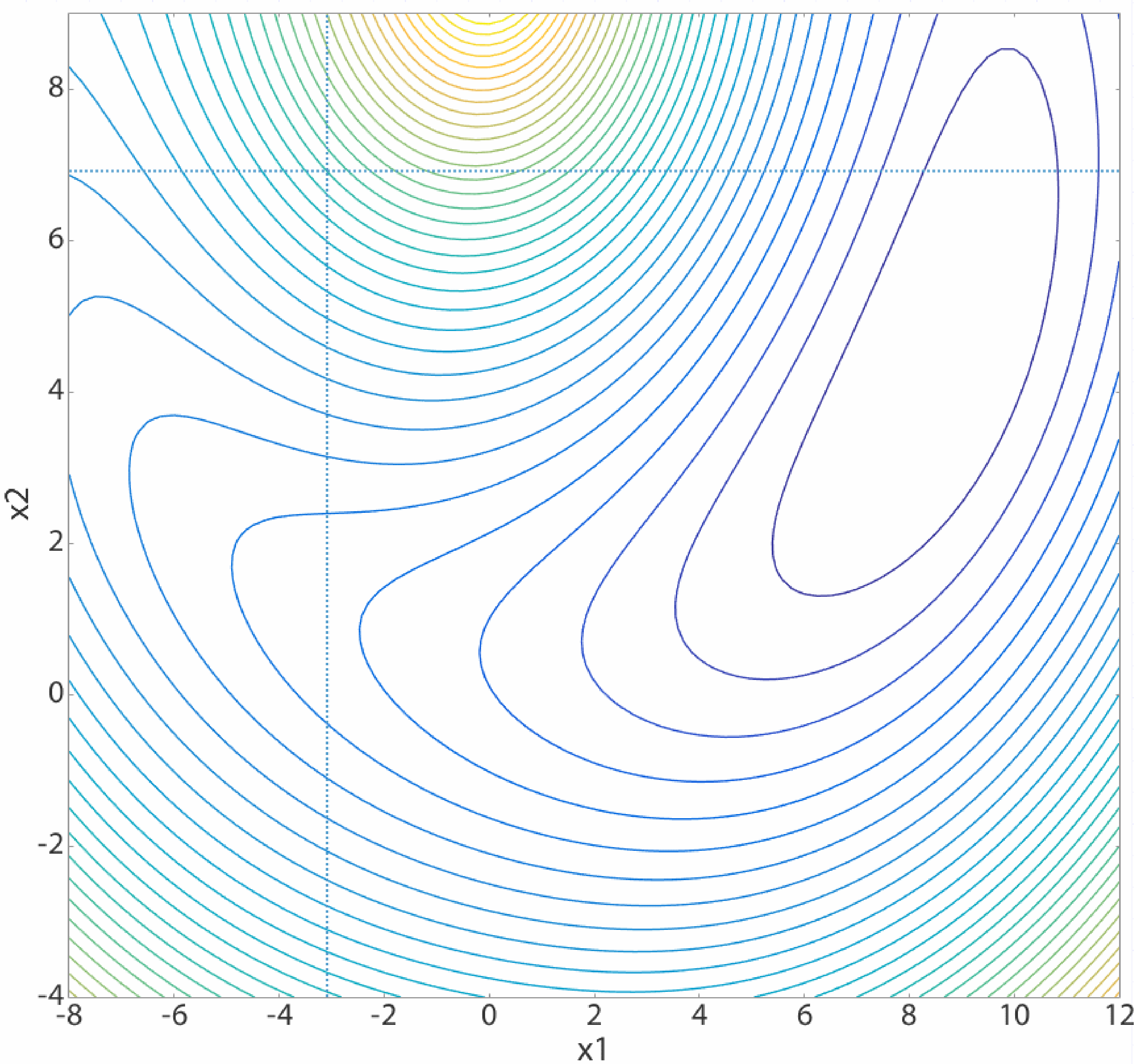


From Rao, S., *Engineering Optimization: Theory and Practice*, 3rd Edition, Wiley Interscience, 1996.

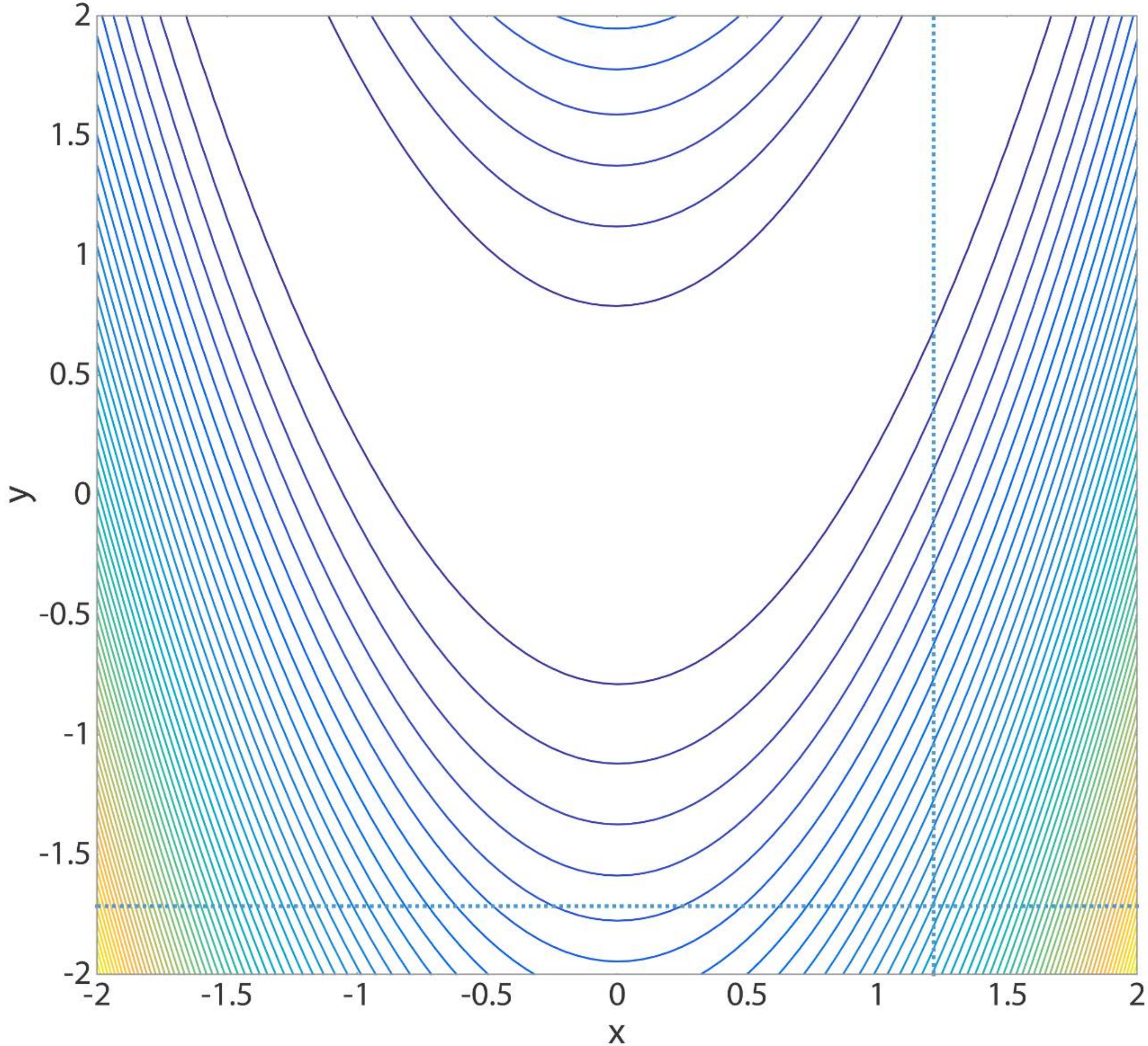
Powell's Method



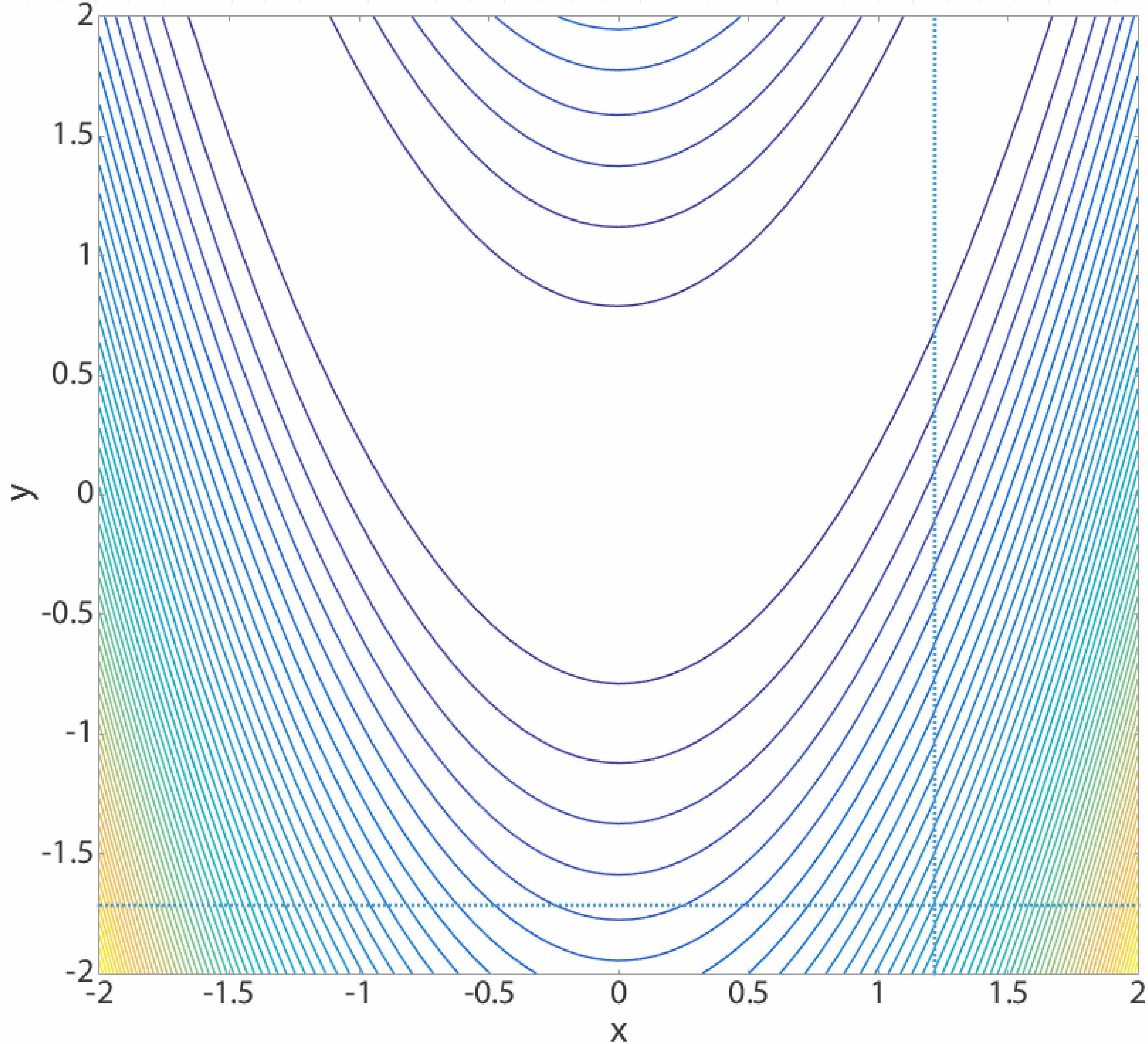
Powell’s Method



Powell's Method



Powell’s Method



Caveats on Powell's Method Convergence

- For functions that are not positive definite quadratics, method will require more iterations or will not converge at all
- It will take additional iterations if the line search method does not find the "exact" minimum along each line
- If the conjugate directions become linearly dependent (parallel), the method can break down; hence, the need to reset the search vectors to the univariate directions every few steps

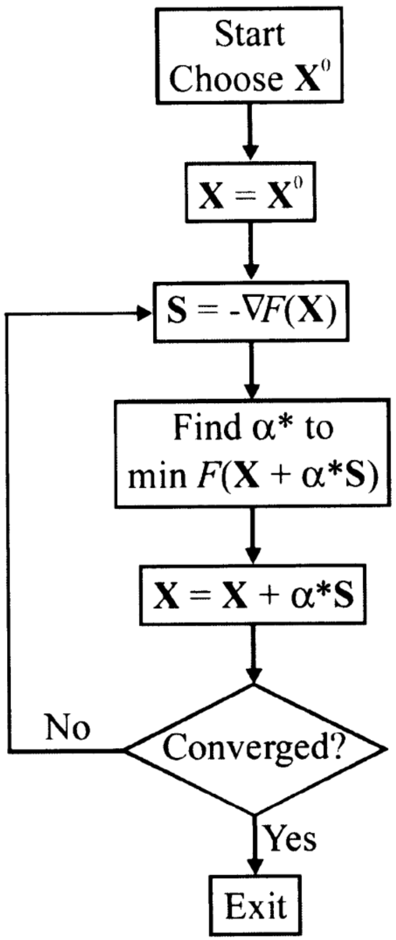
Steepest Descent

The simplest first-order method is steepest descent in which the search direction is chosen as,

$$\boldsymbol{s_k} = -\nabla f_{k-1}$$

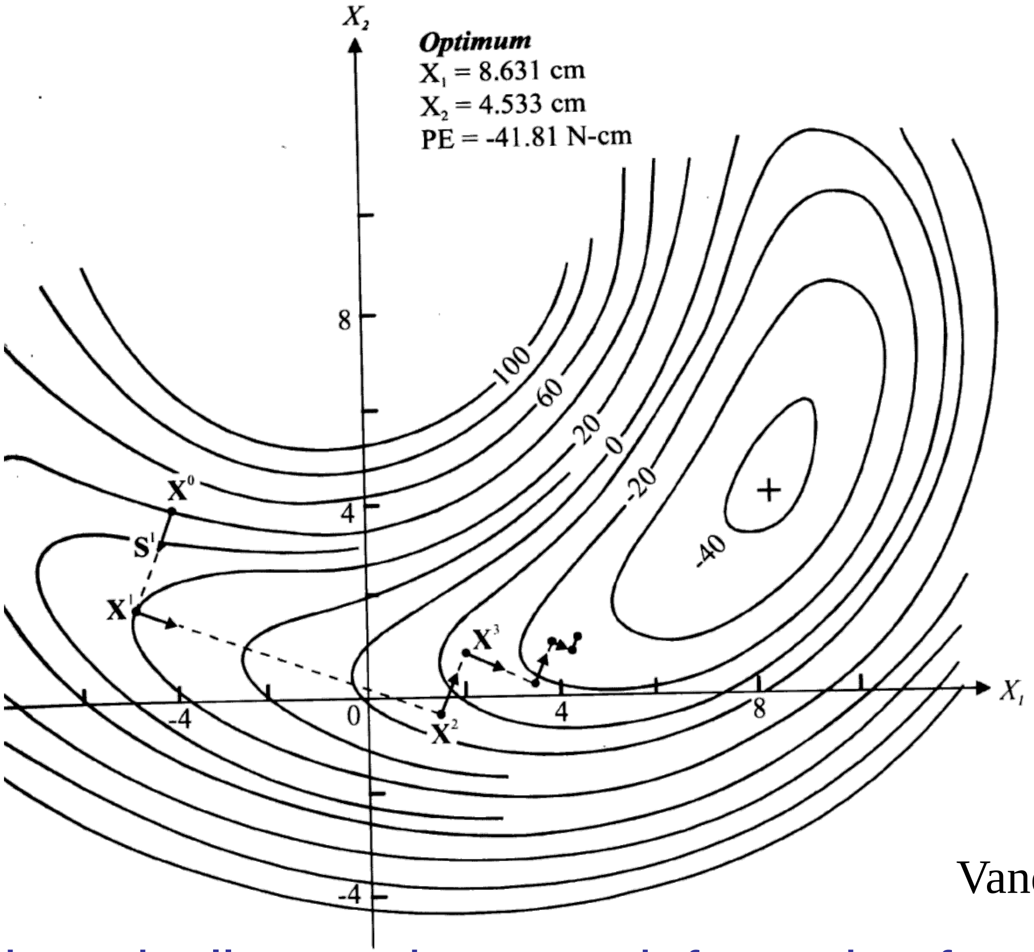
In some implementations, the search direction is normalized as,

$$\boldsymbol{s_k} = -\frac{\nabla f_{k-1}}{||\nabla f_{k-1}||}$$



Vanderplaats, Fig. 3.6

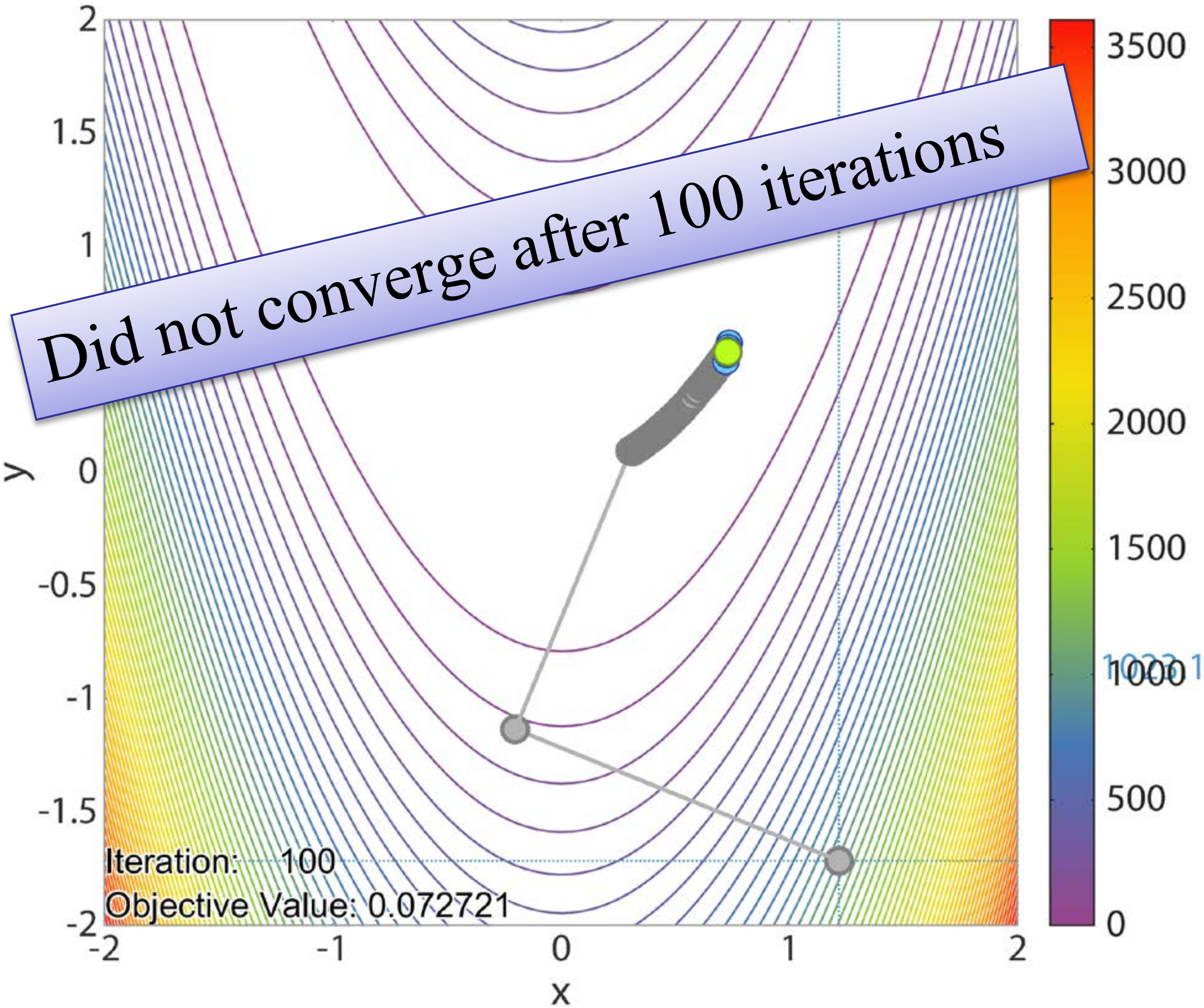
Steepest Descent



Vanderplaats, Fig. 3.7

Convergence is typically poor because information from previous search directions is not used in forming new search directions.

Steepest Descent



Fletcher-Reeves Conjugate Gradient Method

As its name implies, the conjugate gradient method creates search directions in which gradient information is used to build conjugate directions. In other words, conjugacy “modifies” a steepest descent search.

This is in contrast to Powell’s method, in which conjugacy “modifies” a univariate search.

Conjugate Gradient Method

The conjugate gradient method defines a search direction based on the following relation:

$$\boldsymbol{s_k} = -\nabla f_{k-1} + \beta_k \boldsymbol{s_{k-1}}$$

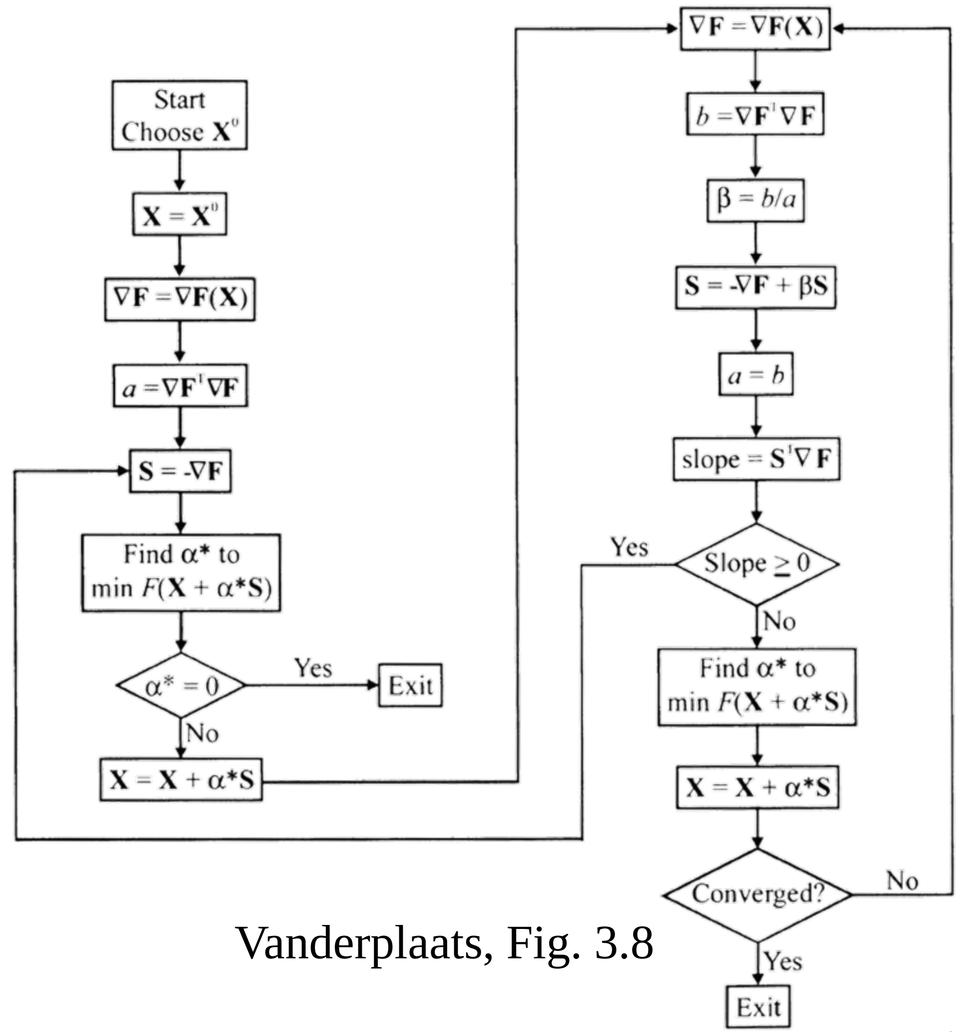
This formulation adds a “correction” to the negative gradient from the previous design point.

The value β_k is not arbitrary; it is chosen to enforce conjugacy of subsequent search directions with respect to the Hessian matrix.

Conjugate Gradient Method

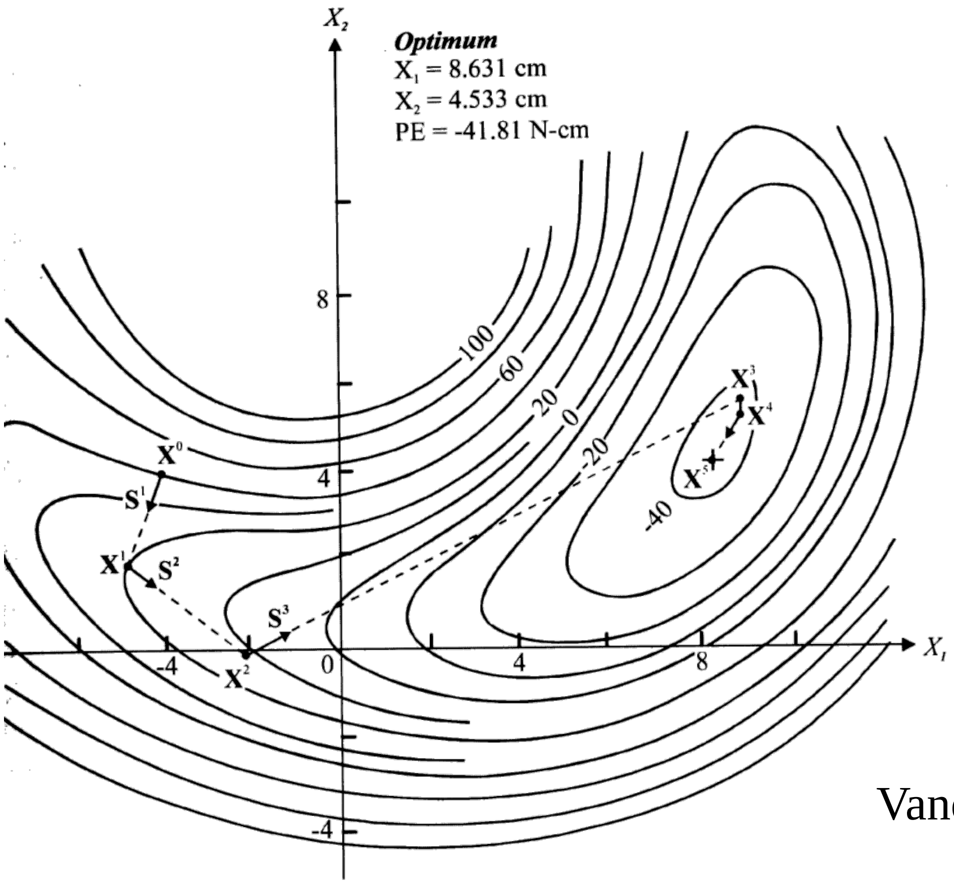
β_k is selected as,

$$\beta_k = \frac{\nabla f_{k-1}^T \nabla f_{k-1}}{\nabla f_{k-2}^T \nabla f_{k-2}}$$



Vanderplaats, Fig. 3.8

Conjugate Gradient Method



Vanderplaats, Fig. 3.9

Convergence is much-improved compared to steepest descent

Conjugate Gradient Method

To understand how β_k is determined, we begin by noting that the conjugacy requirement for the conjugate gradient method is of the form,

$$\boldsymbol{s}_{\boldsymbol{k-1}}^T H_{k-2} \boldsymbol{s}_{\boldsymbol{k}} = 0$$

To enforce this condition, we multiply our search direction equation by $\boldsymbol{s}_{\boldsymbol{k-1}}^T H_{k-2}$ and set the result equal to zero:

$$\boldsymbol{s}_{\boldsymbol{k-1}}^T H_{k-2} [-\nabla f_{k-1} + \beta_k \boldsymbol{s}_{\boldsymbol{k-1}}] = 0$$

Conjugate Gradient Method

Let's now presume that $k = 2$, i.e. we are forming the first conjugate direction.

Additionally, let's presume that the initial search direction was the steepest descent direction, i.e. $\boldsymbol{s_1} = -\nabla f_0$.

Applying these conditions gives,

$$-\nabla f_0^T H_0[-\nabla f_1 - \beta_2 \nabla f_0] = 0$$

Conjugate Gradient Method

We do not know the Hessian directly since this is a first-order method, but we can estimate it in a similar way as we discussed in the development of quasi-Newton methods.

Recall our Taylor series expansion for the gradient,

$$\nabla f_1 = \nabla f_0 + H_0[\boldsymbol{x_1} - \boldsymbol{x_0}]$$

But,

$$\boldsymbol{x_1} = \boldsymbol{x_0} + \alpha_1^* \boldsymbol{s_1} \Rightarrow \boldsymbol{x_1} - \boldsymbol{x_0} = \alpha_1^* \boldsymbol{s_1}$$

Conjugate Gradient Method

So,

$$H_0\, \boldsymbol{s}_1 = [\nabla f_1 - \nabla f_0]/\alpha_1^*$$

Taking the transpose of this result gives,

$$\boldsymbol{s}_1^T H_0^T = \left[\nabla f_1^T - \nabla f_0^T\right]/\alpha_1^*$$

But the Hessian is symmetric, so $H_0 = H_0^T$ and $\boldsymbol{s}_1 = -\nabla f_0$,

$$-\nabla f_0^T H_0 = \left[\nabla f_1^T - \nabla f_0^T\right]/\alpha_1^*$$

Conjugate Gradient Method

We now substitute this result for the Hessian,

$$-\nabla f_0^T H_0 = \left[\nabla f_1^T - \nabla f_0^T\right]/\alpha_1^*$$

into the conjugacy condition,

$$-\nabla f_0^T H_0[-\nabla f_1 - \beta_2 \nabla f_0] = 0$$

to obtain,

$$\left[\nabla f_1^T - \nabla f_0^T\right][\nabla f_1 + \beta_2 \nabla f_0] = 0.$$

Conjugate Gradient Method

$$[\nabla f_1^T - \nabla f_0^T][\nabla f_1 + \beta_2 \nabla f_0] = 0$$

Expanding terms,

$$\nabla f_1^T \nabla f_1 + \beta_2 \nabla f_1^T \nabla f_0 - \nabla f_0^T \nabla f_1 - \beta_2 \nabla f_0^T \nabla f_0 = 0$$

Next, note that $\nabla f_1^T \nabla f_0 = 0$ because ∇f_0 is the initial search direction and the gradient of the function at the new point $\boldsymbol{x_1}$ must be perpendicular to this direction, else we would not have found the minimum in the line search.

Conjugate Gradient Method

The result therefore simplifies to,

$$\nabla f_1^T \nabla f_1 - \beta_2 \nabla f_0^T \nabla f_0 = 0$$

Or,

$$\beta_2 = \frac{\nabla f_1^T \nabla f_1}{\nabla f_0^T \nabla f_0}$$

This result can be shown to generalize for arbitrary *k* to,

$$\beta_k = \frac{\nabla f_{k-1}^T \nabla f_{k-1}}{\nabla f_{k-2}^T \nabla f_{k-2}}$$

Newton's Method

Newton's Method is based on the concept of the Newton search direction, which we discussed earlier.

We derive the Newton direction by considering a second-order Taylor series of the form,

$$f(\boldsymbol{x}_{k-1} + \boldsymbol{s}_k) = f(\boldsymbol{x}_{k-1}) + \boldsymbol{s}_k^T \nabla f(\boldsymbol{x}_{k-1}) + \frac{1}{2}\boldsymbol{s}_k^T H(\boldsymbol{x}_{k-1})\,\boldsymbol{s}_k$$

This is a quadratic approximation in the neighborhood of $\boldsymbol{x}_{k-1}$. Note that α does not appear; this implies that $\alpha = 1$.

Newton's Method

Next, take the derivative of the function with respect to $\boldsymbol{s_k}$ and set it equal zero,

$$\frac{df(\boldsymbol{x_{k-1}} + \boldsymbol{s_k})}{d\boldsymbol{s_k}} = \nabla f(\boldsymbol{x_{k-1}}) + H(\boldsymbol{x_{k-1}})\,\boldsymbol{s_k} = \boldsymbol{0}$$

We can then solve for $\boldsymbol{s_k}$ to obtain,

$$\boldsymbol{s_k} = -[H(\boldsymbol{x_{k-1}})]^{-1}\,\nabla f(\boldsymbol{x_{k-1}})$$

This $\boldsymbol{s_k}$ is called the **Newton search direction.**

Newton's Method

When $H(\boldsymbol{x_{k-1}})$ is positive definite, the Newton direction is a descent direction.

When $H(\boldsymbol{x_{k-1}})$ is not positive definite, we can have two problems:

- $[H(\boldsymbol{x_{k-1}})]^{-1}$ may not exist (this happens when the function is linear in any one variable)
- Even if $[H(\boldsymbol{x_{k-1}})]^{-1}$ exists, the Newton direction may not define a descent direction

A Newton's method algorithm should incorporate ways to modify $\boldsymbol{s_k}$ if these problems occur.

Newton's Method Algorithm

1. Select an initial point, $\boldsymbol{x_0}$ and set $k = 1$

2. Compute H_{k-1} and ∇f_{k-1} either analytically or with finite differences

3. Find the search direction either by inversion of H_{k-1} as $\boldsymbol{s_k} = -[H_{k-1}]^{-1} \nabla f_{k-1}$, or by solving the linear system $H_{k-1}\boldsymbol{s_k} = -\nabla f_{k-1}$ with a technique such as Gaussian elimination.

4. If H_{k-1} is singular, choose an alternate approach to set the search direction. For example, use steepest descent, and set $\boldsymbol{s_k} = -\nabla f_{k-1}$.

Newton's Method Algorithm

5. Find α^* along the Newton direction $\boldsymbol{s_k}$ and move to this point. A very good guess is $\alpha^* = 1$, which is the exact solution if the function is quadratic. There are two choices for how to do this:
   a. Set $\alpha^* = 1$ and just move to this point. This is the typical approach.
   b. Do a search along $\boldsymbol{s_k}$ with $\alpha^* = 1$ as an initial guess. Algorithms that do this sometimes impose "move limits" that set $\alpha_{move} = \min(\alpha^*, \alpha_{max})$, where α_{max} is specified. Because H_{k-1} is a "local" approximation, this limits problems associated with overshoot and oscillation.

6. Set $k = k + 1$ and repeat steps 2 through 5 until a convergence criterion is met. In some implementations, H_{k-1} is computed only every few iterations in order to reduce computational cost.

Broyden, Fletcher, Goldfarb, Shanno (BFGS) Method

The BFGS method is a "quasi-Newton" method. Quasi-Newton methods are also called "variable metric" methods.

Recall from our discussion of search directions that quasi-Newton methods work by approximating a $B_k \approx H_{k-1}$ that satisfies,

$$B_k \boldsymbol{p}_{\boldsymbol{k}-\mathbf{1}} = \boldsymbol{y}_{\boldsymbol{k}-\mathbf{1}}$$

$$\boldsymbol{p}_{\boldsymbol{k}-\mathbf{1}} = \boldsymbol{x}_{\boldsymbol{k}} - \boldsymbol{x}_{\boldsymbol{k}-\mathbf{1}}$$

$$\boldsymbol{y}_{\boldsymbol{k}-\mathbf{1}} = \nabla f_k - \nabla f_{k-1}$$

The search direction is then found as,

$$\boldsymbol{s}_{\boldsymbol{k}} = -B_k^{-1} \nabla f_{k-1}$$

Broyden, Fletcher, Goldfarb, Shanno (BFGS) Method

Let's define a matrix $\widetilde{H} = B_k^{-1}$. Note that this matrix looks like the inverse of the Hessian, not the Hessian itself.

The BFGS method defines $\widetilde{H}$ directly as follows:

$$\widetilde{H}^{k+1} = \widetilde{H}^k + D^k$$

where D^k is an update matrix of the form,

$$D^k = \frac{\sigma + \tau}{\sigma^2}\boldsymbol{p}\boldsymbol{p}^T - \frac{1}{\sigma}\left[\widetilde{H}^k\boldsymbol{y}\boldsymbol{p}^T + \boldsymbol{p}\left(\widetilde{H}^k\boldsymbol{y}\right)^T\right]$$

Broyden, Fletcher, Goldfarb, Shanno (BFGS) Method

And the scalars σ and τ are defined as,

$$\sigma = \boldsymbol{p}^T \boldsymbol{y}$$

and

$$\tau = \boldsymbol{y}^T \widetilde{H}^k \boldsymbol{y}.$$

Based on this update procedure, the search direction is found as

$$\boldsymbol{s_k} = -\widetilde{H}^k \nabla f_{k-1}$$

Set $\widetilde{H}^1 = I$ (identity matrix) such that the initial search direction is steepest descent.

Other aspects of the search follow similarly to Newton's method.