IE 203 PS 6

Unconstrained Nonlinear Optimization

- Analytic Methods: Finding stationary points with derivative. Checking shape of the function over its domain or at a stationary point, by using second derivative if univariate or Hessian Matrix if multivariate.
- Numerical Methods
  - Univariate
    - Bisection Method
    - Newton's "
    - Secant "
    - ⋮
  - Multivariate
    - Cyclic Coordinate Search
    - Hooke - Jeeves Method
    - ⋮

    No gradient

    - Steepest Descent
    - Newton's Method
    - ⋮

    with gradient

Iterative Search

$$x_{k+1} = x_k + \alpha_k \cdot d_k$$

next point, current point, step size, direction

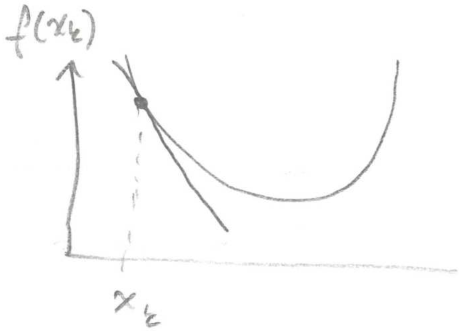
- Bisection Search Method
- Dichotomous Search
- Golden Section Search
- Newton's Method
- Secant Method

(Useful when f is not differentiable) requires a unimodel function/interval

<u>General Iterative Search</u>: $x_{k+1} = x_k + \alpha_k d_k$

step size direction

An idea for a direction: - derivative / - gradient



$$x_{k+1} = x_k + \underbrace{t}_{\text{movement} = \text{step size} \times \text{direction}}$$

<u>Second-Order Taylor Expansion</u>:

$$\underbrace{f(x_k+t)} \approx f(x_k) + f'(x_k)t + \frac{1}{2!} f''(x_k)t^2$$

value of function at next iteration, which we want to minimize

$$\frac{d}{dt}\left(f(x_k) + f'(x_k)t + \frac{1}{2} f''(x_k)t^2\right)$$

$$= f'(x_k) + f''(x_k)t$$

Set equal to zero: $f'(x_k) + f''(x_k)t = 0$

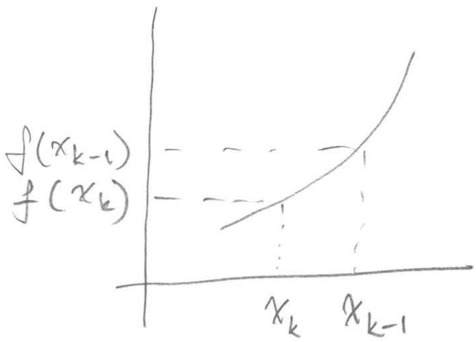
$$t = -\frac{f'(x_k)}{f''(x_k)}$$

Then newton's method:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

$$\lim_{h \to 0} f'(x+h) = \frac{f(x+h) - f(x)}{h}$$

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$



* In optimization, we want the roots of $f'(x) = 0$.

Newton's Method:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Secant Method:

$$x_{k+1} = x_k - f'(x_k) \frac{x_k - x_{k-1}}{f'(x_k) - f'(x_{k-1})}$$

Multivariate Search

Choosing a step size

Exact (Accurate) Line Search

Say f is a multivariate function, $f(x_1, x_2, \ldots, x_n)$. Say we have a current point $x_k = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ and a direction $d_k = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$ if we choose $\alpha_k = \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} f(x_k + \alpha d_k)$ we will obtain best x_{k+1} that can be achieved using x_k and d_k in one iteration.

Example: $f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$

$$x_1 = (0, 3)$$
$$d_1 = (1, 0)$$

$$f(x_1 + \alpha d_1) = f\big((0,3) + \alpha(1,0)\big) = f(\alpha, 3) = (\alpha - 2)^4 + (\alpha - 6)^2$$

$f_\alpha(\alpha) = (\alpha-2)^4 + (\alpha-6)^2$ → univariate fnc. solve analytically if easy, otherwise use (bisection/Newton's/Secant)

→ use desmos.com/calculator to find the min of $f_\alpha(\alpha)$

→ $$\alpha_k = \underset{\alpha \in \mathbb{R}}{\arg\min}\ f_\alpha(\alpha) = 3.13$$

Inaccurate Line Search

If finding the best step size (α) is costly, how about we choose the step size some other way?

Choose a series $\{a_k\}_{k=0}^{\infty}$ that monotonically decreases.

Should the series be convergent ($\lim_{k\to\infty} \sum_{j=1}^{k} a_j = c$)

or divergent ($\lim_{k\to\infty} \sum_{j=1}^{k} a_j \to \infty$)?

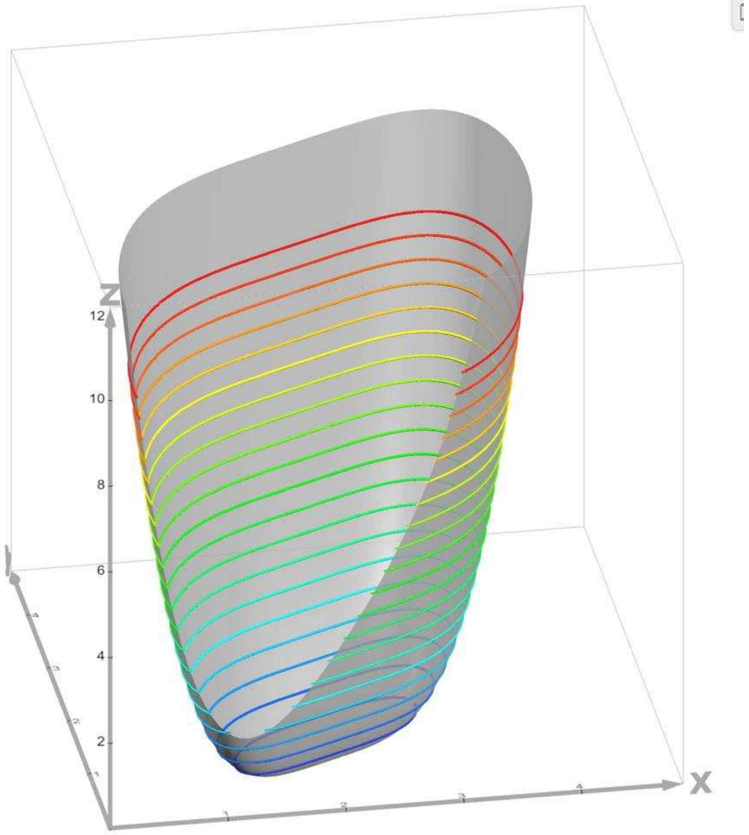
An example series: $a_k = \frac{1}{k}$ (Harmonic series)

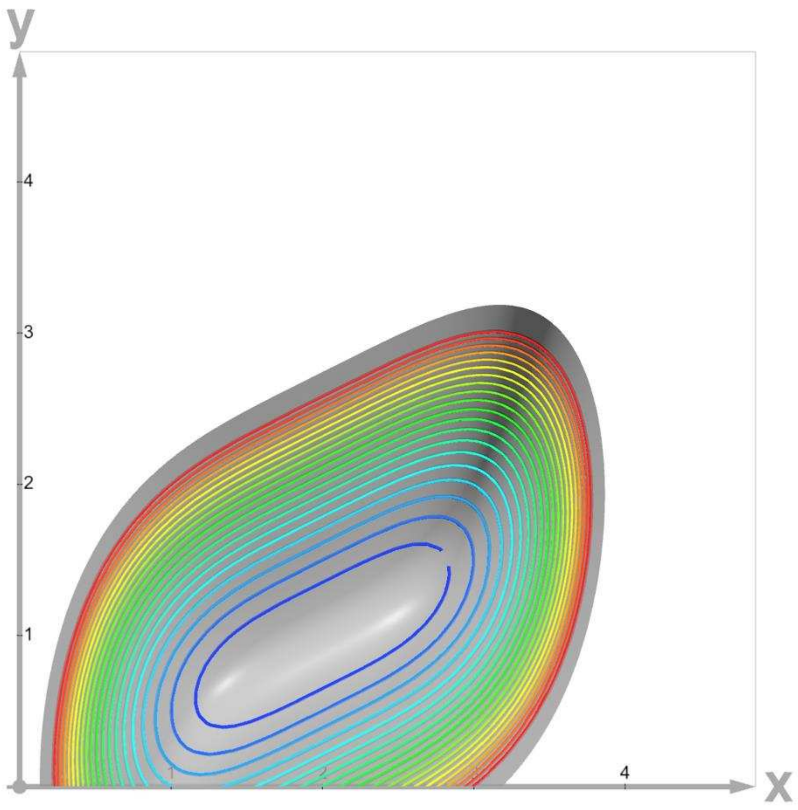
See why $\alpha_k = \frac{1}{2^k}$ would be a bad choice:

Let $f(x) = x^2$, $x_1 = 2$, $d_k = -1$ $\forall k \in \mathbb{Z}_{+,0}$. Can you reach $x = 0$?

Example: $f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$

$$x^* = \underset{x \in \mathbb{R}}{argmin}\, f(x_1, x_2) = (2, 1)$$





https://www.desmos.com/3d/6kqdio2lmw

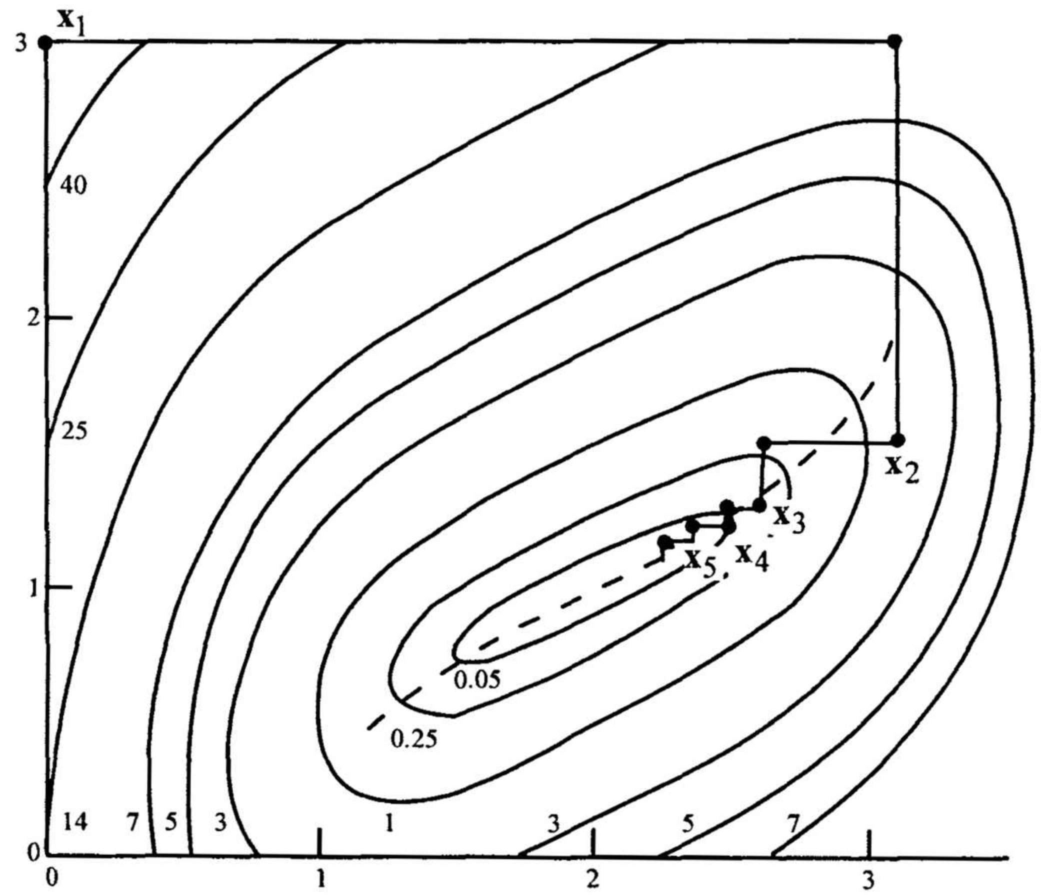
Cyclic Coordinate Search

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$$
$$x_1 = (0,3)$$

Initialization Step Choose a scalar $\varepsilon > 0$ to be used for terminating the algorithm, and let $\mathbf{d}_1, \ldots, \mathbf{d}_n$ be the coordinate directions. Choose an initial point $\mathbf{x}_1$, let $\mathbf{y}_1 = \mathbf{x}_1$, let $k = j = 1$, and go to the Main Step.

Main Step

1. Let λ_j be an optimal solution to the problem to minimize $f(\mathbf{y}_j + \lambda \mathbf{d}_j)$ subject to $\lambda \in R$, and let $\mathbf{y}_{j+1} = \mathbf{y}_j + \lambda_j \mathbf{d}_j$. If $j < n$, replace j by $j + 1$, and repeat Step 1. Otherwise, if $j = n$, go to Step 2.
2. Let $\mathbf{x}_{k+1} = \mathbf{y}_{n+1}$. If $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon$, then stop. Otherwise, let $\mathbf{y}_1 = \mathbf{x}_{k+1}$, let $j = 1$, replace k by $k + 1$, and go to Step 1.



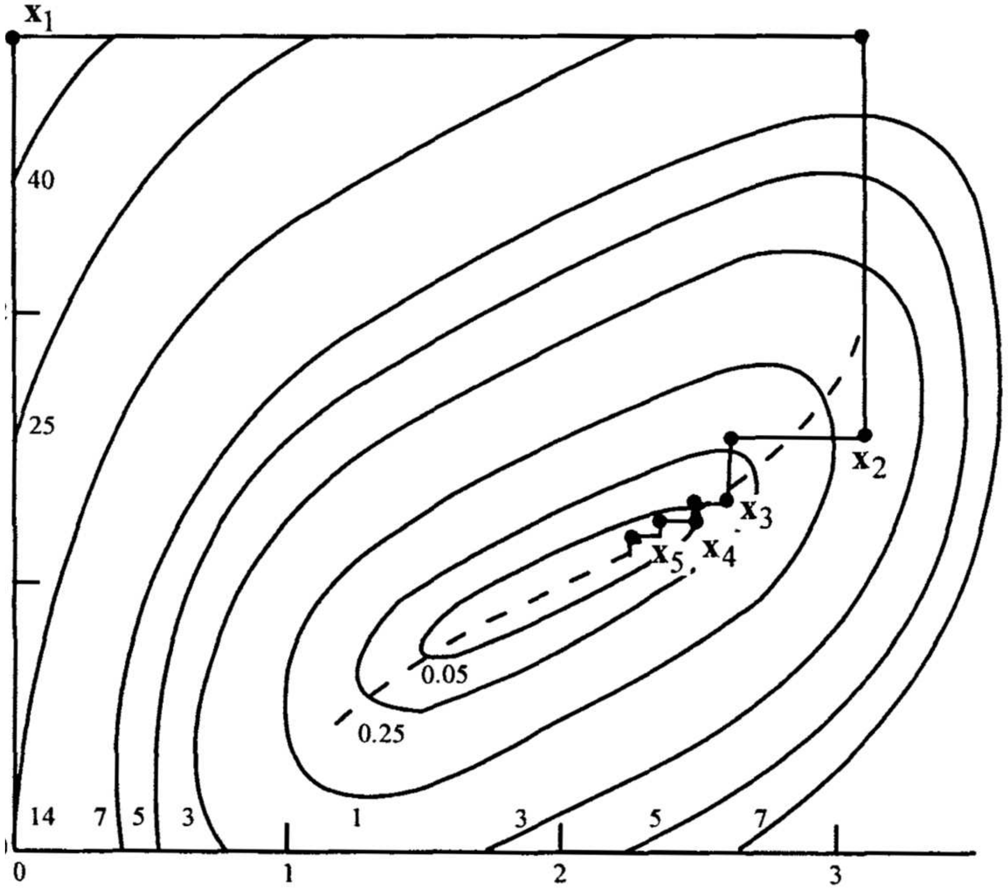
Reference: M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd Edition, 2006

Cyclic Coordinate Search

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$$
$$x_1 = (0,3)$$

Table 8.6 Summary of Computations for the Cyclic Coordinate Method

Iteration k	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	j	$\mathbf{d}_j$	$\mathbf{y}_j$	λ_j	$\mathbf{y}_{j+1}$
1	(0.00, 3.00)	1	(1.0, 0.0)	(0.00,3.00)	3.13	(3.13, 3.00)
	52.00	2	(0.0, 1.0)	(3.13,3.00)	–1.44	(3.13, 1.56)
2	(3.13, 1.56)	1	(1.0, 0.0)	(3.13, 1.56)	–0.50	(2.63, 1.56)
	1.63	2	(0.0, 1.0)	(2.63, 1.56)	–0.25	(2.63, 1.31)
3	(2.63, 1.31)	1	(1.0, 0.0)	(2.63, 1.31)	–0.19	(2.44, 1.31)
	0.16	2	(0.0, 1.0)	(2.44, 1.31)	–0.09	(2.44, 1.22)
4	(2.44, 1.22)	1	(1.0, 0.0)	(2.44, 1.22)	–0.09	(2.35, 1.22)
	0.04	2	(0.0, 1.0)	(2.35, 1.22)	–0.05	(2.35, 1.17)
5	(2.35, 1.17)	1	(1.0, 0.0)	(2.35, 1.17)	–0.06	(2.29, 1.17)
	0.015	2	(0.0, 1.0)	(2.29, 1.17)	–0.03	(2.29, 1.14)
6	(2.29, 1.14)	1	(1.0, 0.0)	(2.29, 1.14)	–0.04	(2.25, 1.14)
	0.007	2	(0.0, 1.0)	(2.25, 1.14)	–0.02	(2.25, 1.12)
7	(2.25, 1.12)	1	(1.0, 0.0)	(2.25, 1.12)	–0.03	(2.22, 1.12)
	0.004	2	(0.0, 1.0)	(2.22,1.12)	–0.01	(2.22, 1.11)



Reference: M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd Edition, 2006

Cyclic Coordinate Search

Stalling

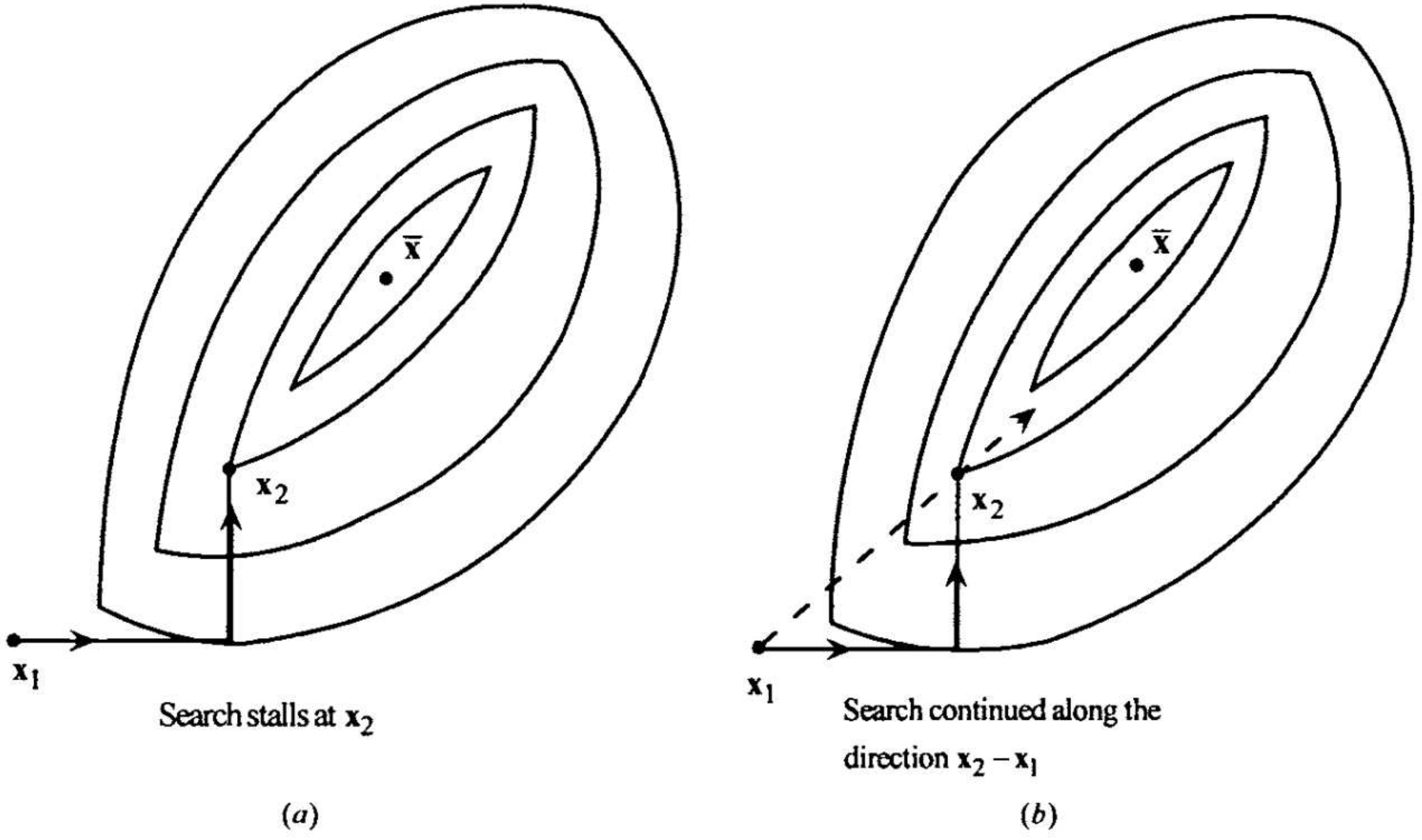


Figure 8.8 Effect of a sharp-edged valley.

Reference: M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd Edition, 2006

Hook-Jeeves Method

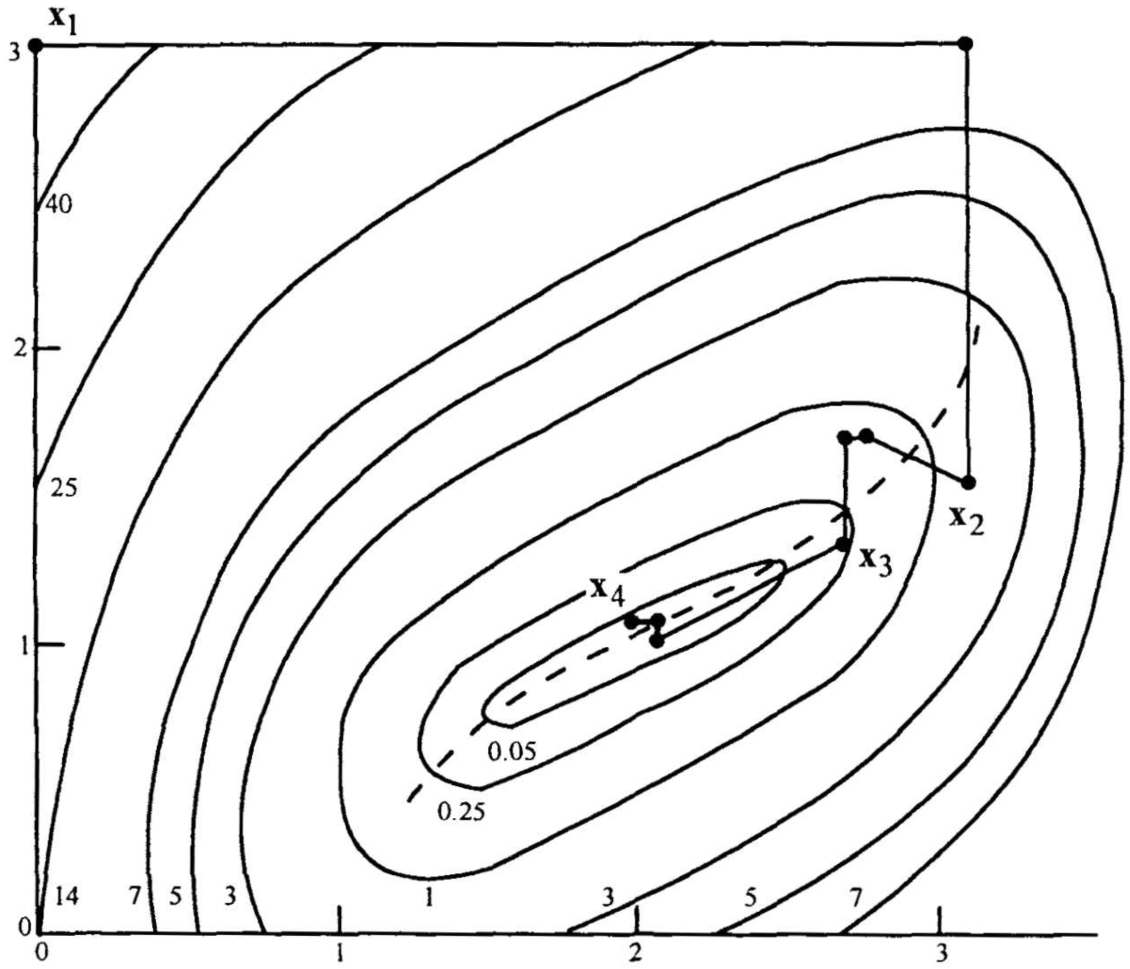
$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$$
$$x_1 = (0,3)$$

Initialization Step Choose a scalar $\varepsilon > 0$ to be used in terminating the algorithm. Choose a starting point $\mathbf{x}_1$, let $\mathbf{y}_1 = \mathbf{x}_1$, let $k = j = 1$, and go to the Main Step.

Main Step

1. Let λ_j be an optimal solution to the problem to minimize $f(\mathbf{y}_j + \lambda \mathbf{d}_j)$ subject to $\lambda \in R$, and let $\mathbf{y}_{j+1} = \mathbf{y}_j + \lambda_j \mathbf{d}_j$. If $j < n$, replace j by $j + 1$, and repeat Step 1. Otherwise, if $j = n$, let $\mathbf{x}_{k+1} = \mathbf{y}_{n+1}$. If $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon$, stop; otherwise, go to Step 2.

2. Let $\mathbf{d} = \mathbf{x}_{k+1} - \mathbf{x}_k$, and let $\hat{\lambda}$ be an optimal solution to the problem to minimize $f(\mathbf{x}_{k+1} + \lambda \mathbf{d})$ subject to $\lambda \in R$. Let $\mathbf{y}_1 = \mathbf{x}_{k+1} + \hat{\lambda} \mathbf{d}$, let $j = 1$, replace k by $k + 1$, and go to Step 1.



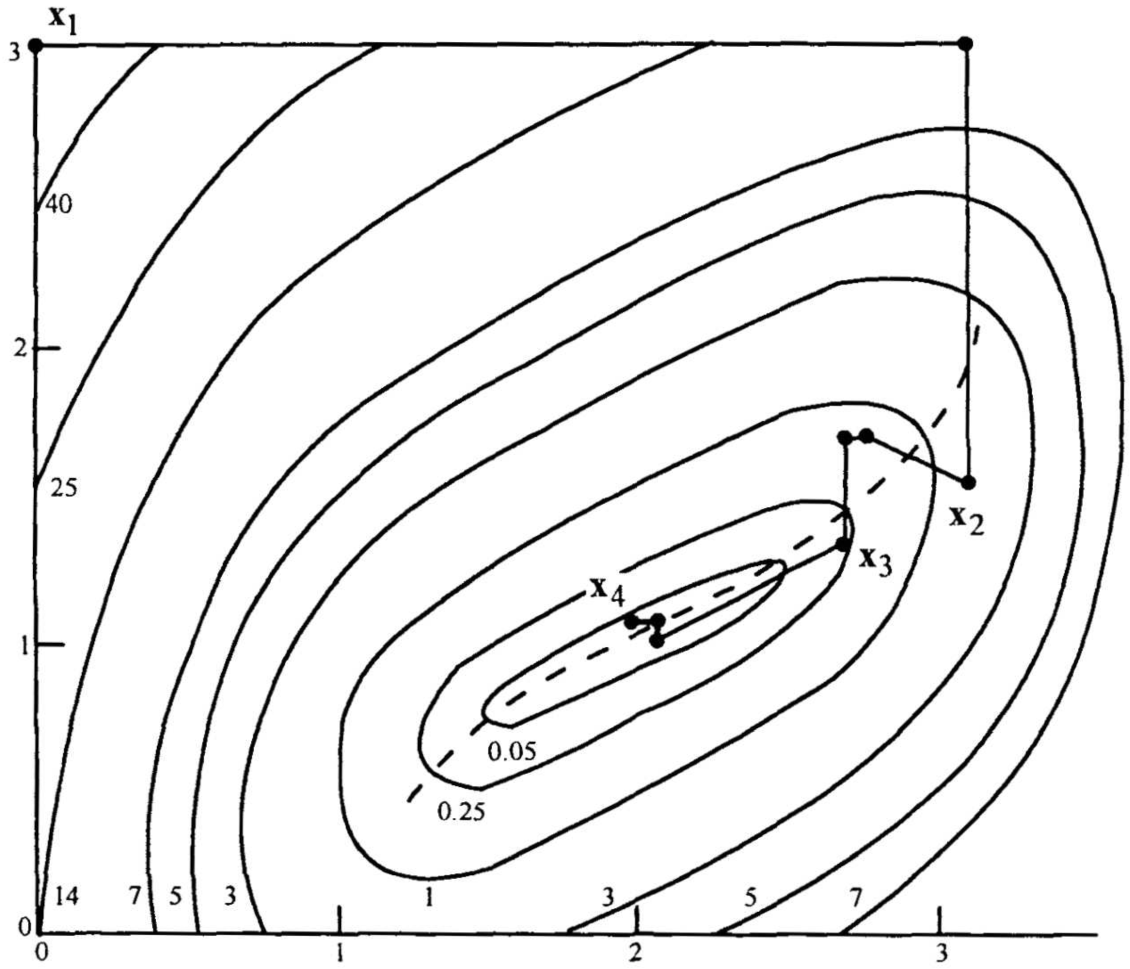
Reference: M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd Edition, 2006

Hook-Jeeves Method

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$$
$$x_1 = (0,3)$$

Table 8.7 Summary of Computations for the Method of Hooke and Jeeves Using Line Searches

Iteration k	$\mathbf{x}_k$ / $f(\mathbf{x}_k)$	j	$\mathbf{y}_j$	$\mathbf{d}_j$	λ_j	$\mathbf{y}_{j+1}$	$\mathbf{d}$	$\hat{\lambda}$	$\mathbf{y}_3 + \hat{\lambda}\mathbf{d}$
1	(0.00, 3.00)	1	(0.00, 3.00)	(1.0, 0.0)	3.13	(3.13, 3.00)	—	—	—
	52.00	2	(3.13, 3.00)	(0.0, 1.0)	–1.44	(3.13, 1.56)	(3.13, 1.44)	–0.10	(2.82, 1.70)
2	(3.13, 1.56)	1	(2.82, 1.70)	(1.0, 0.0)	–0.12	(2.70, 1.70)	—	—	—
	1.63	2	(2.70, 1.70)	(0.0, 1.0)	–0.35	(2.70, 1.35)	(–0.43, –0.21)	1.50	(2.06, 1.04)
3	(2.70, 1.35)	1	(2.06, 1.04)	(1.0, 0.0)	–0.02	(2.04, 1.04)	—	—	—
	0.24	2	(2.04, 1.04)	(0.0, 1.0)	–0.02	(2.04, 1.02)	(–0.66, –0.33)	0.06	(2.00, 1.00)
4	(2.04, 1.02)	1	(2.00, 1.00)	(1.0, 0.0)	0.00	(2.00, 1.00)	—	—	—
	0.000003	2	(2.00, 1.00)	(0.0, 1.0)	0.00	(2.00, 1.00)			
5	(2.00, 1.00)								
	0.00								



Reference: M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd Edition, 2006

Steepest Descent

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$$
$$x_1 = (0,3)$$

Table 8.11 Summary of Computations for the Method of Steepest Descent

Iteration k	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$\nabla f(\mathbf{x}_k)$	$\lVert\nabla f(\mathbf{x}_k)\rVert$	$\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$	λ_k	$\mathbf{x}_{k+1}$
1	(0.00, 3.00) 52.00	(–44.00, 24.00)	50.12	(44.00, –24.00)	0.062	(2.70, 1.51)
2	(2.70, 1.51) 0.34	(0.73, 1.28)	1.47	(–0.73, –1.28)	0.24	(2.52, 1.20)
3	(2.52, 1.20) 0.09	(0.80, –0.48)	0.93	(–0.80, 0.48)	0.11	(2.43, 1.25)
4	(2.43, 1.25) 0.04	(0.18, 0.28)	0.33	(–0.18, –0.28)	0.31	(2.37, 1.16)
5	(2.37, 1.16) 0.02	(0.30, –0.20)	0.36	(–0.30, 0.20)	0.12	(2.33, 1.18)
6	(2.33, 1.18) 0.01	(0.08, 0.12)	0.14	(–0.08, –0.12)	0.36	(2.30, 1.14)
7	(2.30, 1.14) 0.009	(0.15, –0.08)	0.17	(–0.15, 0.08)	0.13	(2.28, 1.15)
8	(2.28, 1.15) 0.007	(0.05, 0.08)	0.09			

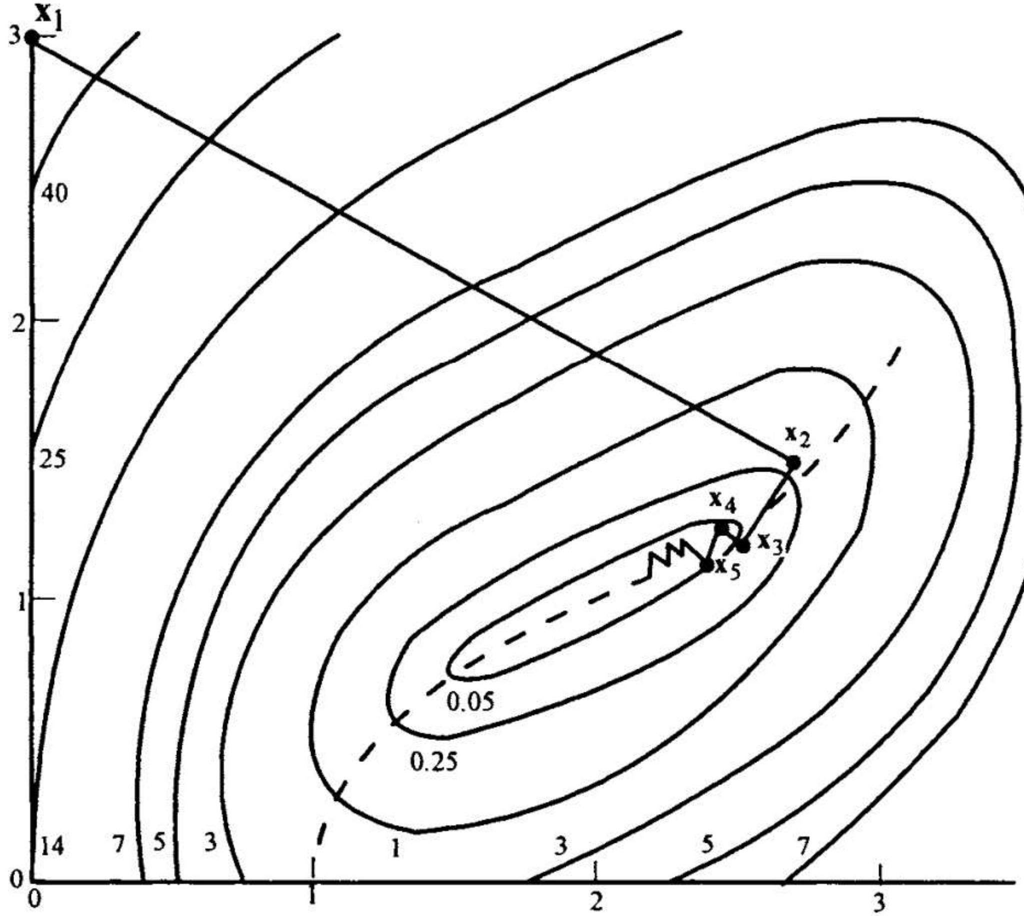


Figure 8.16 Method of steepest descent.

Reference: M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd Edition, 2006

Newton's Method

$$x_{k+1} = x_k - H(x_k)^{-1}\nabla f(x_k)$$

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$$
$$x_1 = (0,3)$$

Table 8.12 Summary of Computations for the Method of Newton

Iteration k	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$\nabla f(\mathbf{x}_k)$	$\mathbf{H}(\mathbf{x}_k)$	$\mathbf{H}(\mathbf{x}_k)^{-1}$	$-\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$	$\mathbf{x}_{k+1}$
1	(0.00, 3.00) 52.00	(–44.0, 24.0)	$\begin{bmatrix} 50.0 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{384}\begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 50.0 \end{bmatrix}$	(0.67, –2.67)	(0.67, 0.33)
2	(0.67, 0.33) 3.13	(–9.39, –0.04)	$\begin{bmatrix} 23.23 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{169.84}\begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 23.23 \end{bmatrix}$	(0.44, 0.23)	(1.11, 0.56)
3	(1.11, 0.56) 0.63	(–2.84, –0.04)	$\begin{bmatrix} 11.50 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{76}\begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 11.50 \end{bmatrix}$	(0.30, 0.14)	(1.41, 0.70)
4	(1.41, 0.70) 0.12	(–0.80, –0.04)	$\begin{bmatrix} 6.18 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{33.44}\begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 6.18 \end{bmatrix}$	(0.20, 0.10)	(1.61, 0.80)
5	(1.61, 0.80) 0.02	(–0.22, –0.04)	$\begin{bmatrix} 3.83 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{14.64}\begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 3.83 \end{bmatrix}$	(0.13, 0.07)	(1.74, 0.87)
6	(1.74, 0.87) 0.005	(–0.07, 0.00)	$\begin{bmatrix} 2.81 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{6.48}\begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 2.81 \end{bmatrix}$	(0.09, 0.04)	(1.83, 0.91)
7	(1.83, 0.91) 0.0009	(0.0003, –0.04)				

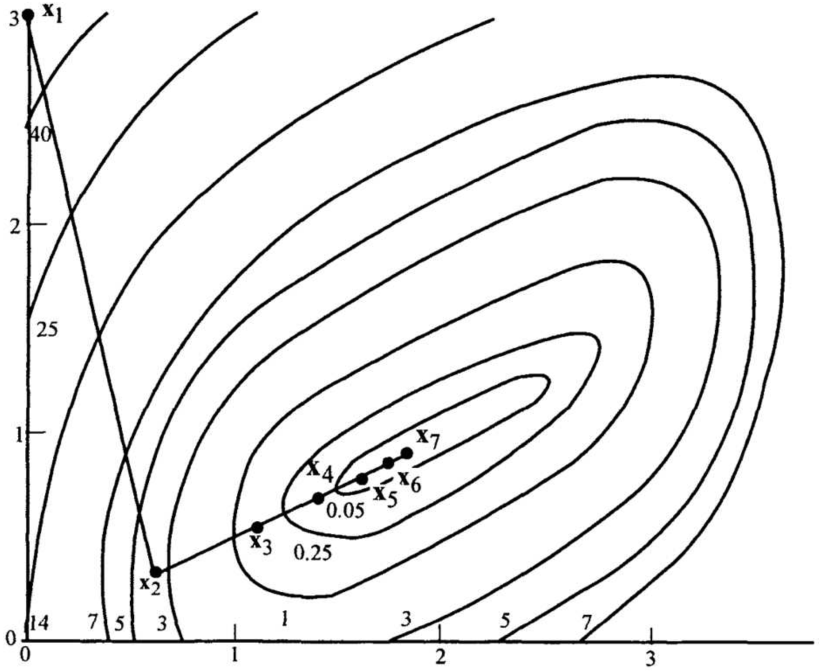


Figure 8.18 Method of Newton.

Reference: M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd Edition, 2006