Topic 5: Non-Linear Regression

- The models we've worked with so far have been *linear in the parameters*.
- They've been of the form: $\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$
- Many models based on economic theory are actually *non-linear* in the parameters.

CES Production function:

$$Y_i = \gamma\left[\delta K_i^{-\rho} + (1-\delta)L_i^{-\rho}\right]^{-\nu/\rho}\exp(\varepsilon_i)$$

or, $$ln(Y_i) = ln(\gamma) - \left(\frac{\nu}{\rho}\right) ln\left[\delta K_i^{-\rho} + (1-\delta)L_i^{-\rho}\right] + \varepsilon_i$$

Linear Expenditure System: (*Stone, 1954*)

Max. $U(\boldsymbol{q}) = \sum_i \beta_i ln(q_i - \gamma_i)$ (*Stone-Geary /Klein-Rubin*)

s.t. $\sum_i p_i q_i = M$

- Yields the following system of demand equations:

  $p_i q_i = \gamma_i p_i + \beta_i\left(M - \sum_j \gamma_j p_j\right)$; $i = 1, 2, \ldots, n$

- The β_i's are the *Marginal Budget Shares*.
- So, we require that $0 < \beta_i < 1$; $i = 1, 2, \ldots, n$.
- Engel aggregation implies that
  1. $\sum_i \gamma_i = 0$.
  2. $\sum_i \beta_i = 1$.
- In general, suppose we have a single non-linear equation:

$$y_i = f\left(x_{i1}, x_{i2}, \ldots, x_{ik}; \theta_1, \theta_2, \ldots, \theta_p\right) + \varepsilon_i$$

- We can still consider a "Least Squares" approach.
- The Non-Linear Least Squares estimator is the vector, $\widehat{\boldsymbol{\theta}}$, that *minimizes* the quantity:

  $S(X, \boldsymbol{\theta}) = \sum_{\boldsymbol{i}}\left[y_i - f_i(X, \widehat{\boldsymbol{\theta}})\right]^{\mathbf{2}}$.

- Clearly the usual LS estimator is just a special case of this.
- To obtain the estimator, we differentiate S with respect to each element of $\widehat{\boldsymbol{\theta}}$; set up the "*p*" first-order conditions and solve.

- Difficulty – usually, the first-order conditions are themselves non-linear in the unknowns (the parameters).
- This means there is (generally) no exact, closed-form, solution.
- Can't write down an explicit formula for the estimators of parameters.

Example

$$y_i = \theta_1 + \theta_2 x_{i2} + \theta_3 x_{i3} + (\theta_2\theta_3)x_{i4} + \varepsilon_i$$

$$S = \sum_i [y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - (\theta_2\theta_3)x_{i4}]^2$$

$$\frac{\partial S}{\partial \theta_1} = -2\sum_i [y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - (\theta_2\theta_3)x_{i4}]$$

$$\frac{\partial S}{\partial \theta_2} = -2\sum_i [(\theta_3 x_{i4} + x_{i2})(y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - \theta_2\theta_3 x_{i4})]$$

$$\frac{\partial S}{\partial \theta_3} = -2\sum_i [(\theta_2 x_{i4} + x_{i3})(y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - \theta_2\theta_3 x_{i4})]$$

Setting these 3 equations to zero, we can't solve analytically for the estimators of the three parameters.

- In situations such as this, we need to use a numerical algorithm to obtain ***a solution*** to the first-order conditions.
- Lots of methods for doing this – one possibility is Newton's algorithm (the **Newton-Raphson algorithm**).

Methods of Descent

$$\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_0 + s\,\boldsymbol{d}(\boldsymbol{\theta}_0)$$

$\boldsymbol{\theta}_0$ = initial (vector) value.

s = step-length (positive scalar)

$\boldsymbol{d}(.)$ = direction vector

- Usually, $\boldsymbol{d}(.)$ Depends on the gradient vector at $\boldsymbol{\theta}_0$.
- It may also depend on the change in the gradient (the Hessian matrix) at $\boldsymbol{\theta}_0$.
- Some specific algorithms in the "family" make the step-length a function of the Hessian.
- One very useful, specific member of the family of "Descent Methods" is the **Newton-Raphson algorithm**:

Suppose we want to minimize some function, $f(\boldsymbol{\theta})$.

Approximate the function using a Taylor's series expansion about $\tilde{\boldsymbol{\theta}}$, the vector value that minimizes $f(\boldsymbol{\theta})$:

$$f(\boldsymbol{\theta}) \cong f(\tilde{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})' \left(\frac{\partial f}{\partial \boldsymbol{\theta}}\right)_{\tilde{\boldsymbol{\theta}}} + \frac{1}{2!}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})' \left[\frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'}\right]_{\tilde{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})$$

Or:

$$f(\boldsymbol{\theta}) \cong f(\tilde{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})' g(\tilde{\boldsymbol{\theta}}) + \frac{1}{2!}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})' H(\tilde{\boldsymbol{\theta}})(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})$$

So,

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cong 0 + (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})' g(\tilde{\boldsymbol{\theta}}) + \frac{1}{2!} 2H(\tilde{\boldsymbol{\theta}})(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})$$

However, $g(\tilde{\boldsymbol{\theta}}) = 0$; as $\tilde{\boldsymbol{\theta}}$ locates a minimum.

So,

$$(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) \cong H^{-1}(\tilde{\boldsymbol{\theta}}) \left(\frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right) \ ;$$

or, $$\tilde{\boldsymbol{\theta}} \cong \boldsymbol{\theta} - H^{-1}(\tilde{\boldsymbol{\theta}}) g(\boldsymbol{\theta})$$

This suggests a numerical algorithm:

Set $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ to begin, and then iterate –

$$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 - H^{-1}(\boldsymbol{\theta}_1)g(\boldsymbol{\theta}_0)$$

$$\boldsymbol{\theta}_2 = \boldsymbol{\theta}_1 - H^{-1}(\boldsymbol{\theta}_2)g(\boldsymbol{\theta}_1)$$

$$\vdots \qquad \vdots \qquad\qquad\qquad \vdots$$

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - H^{-1}(\boldsymbol{\theta}_{n+1})g(\boldsymbol{\theta}_n)$$

or, approximately:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - H^{-1}(\boldsymbol{\theta}_n)g(\boldsymbol{\theta}_n)$$

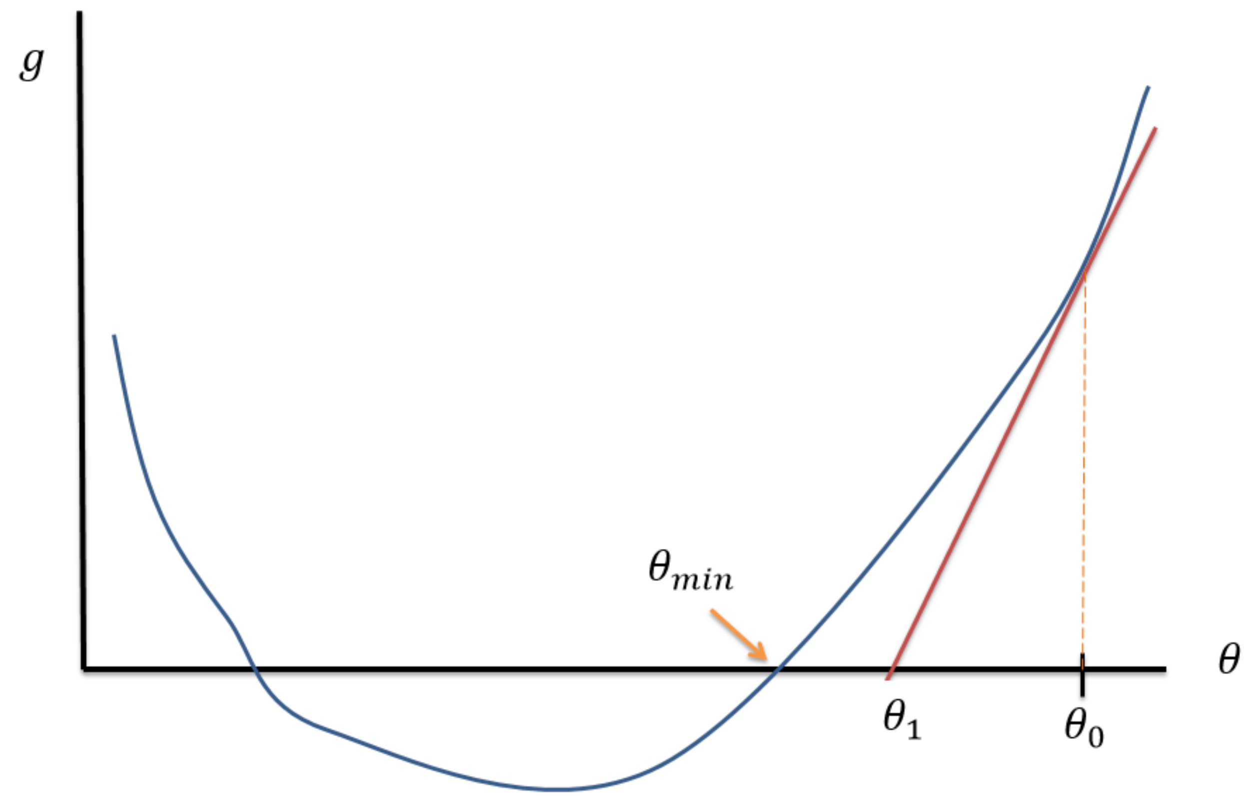
Stop if $\left|\frac{\left(\theta_{n+1}^{(i)} - \theta_n^{(i)}\right)}{\theta_n^{(i)}}\right| < \varepsilon^{(i)} \quad ; \quad i = 1, 2, \ldots, p$

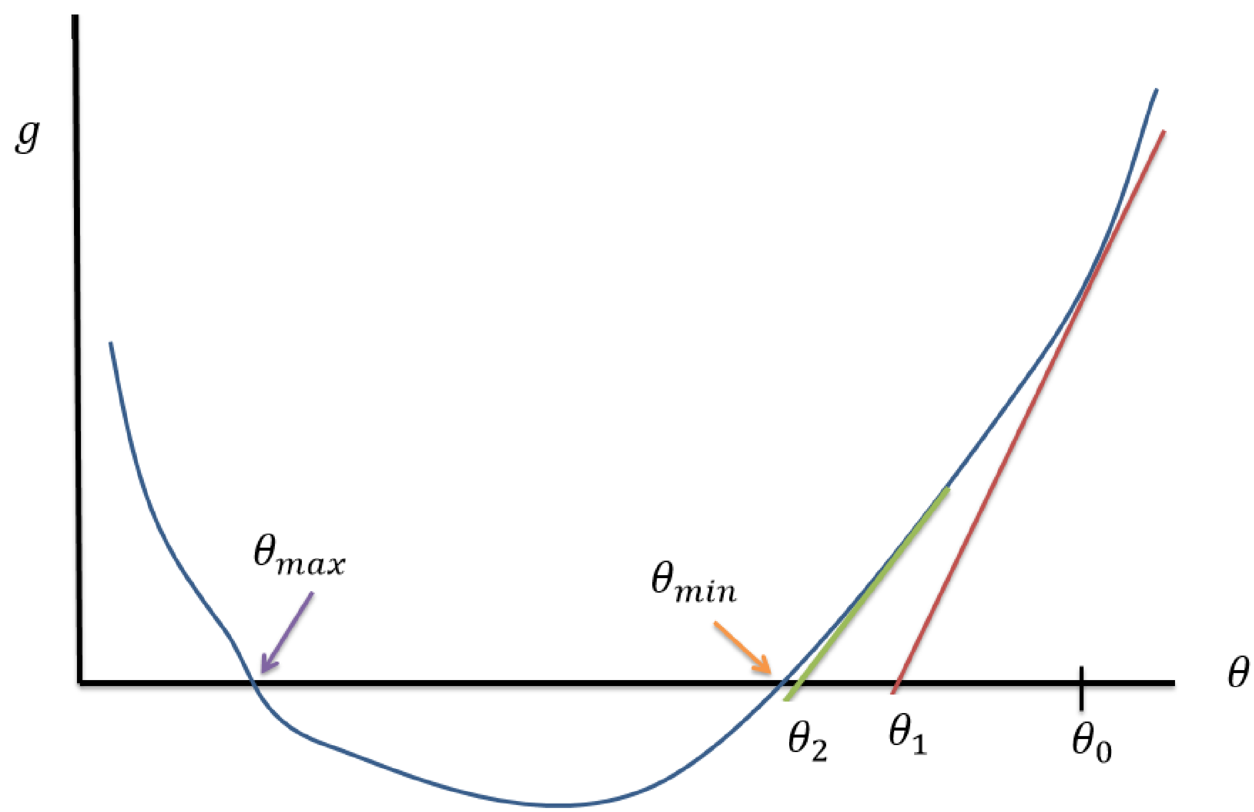
Note:

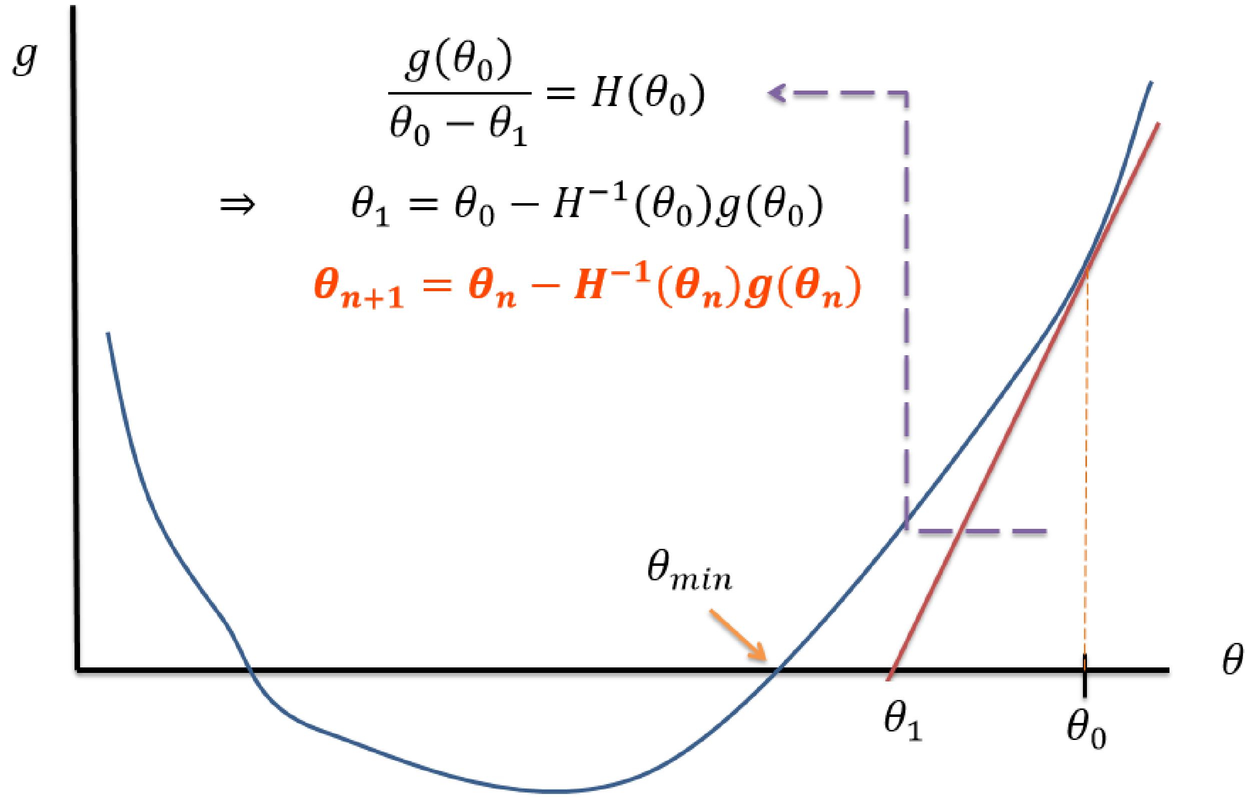
1. $s = 1$.
2. $\boldsymbol{d}(\boldsymbol{\theta}_n) = -H^{-1}(\boldsymbol{\theta}_n)g(\boldsymbol{\theta}_n)$.
3. Algorithm *fails* if H ever becomes *singular* at any iteration.
4. Achieve a *minimum* of f (.) if H is *positive definite*.
5. Algorithm may locate only a *local* minimum.
6. Algorithm may *oscillate*.

The algorithm can be given a nice *geometric interpretation* – scalar θ.

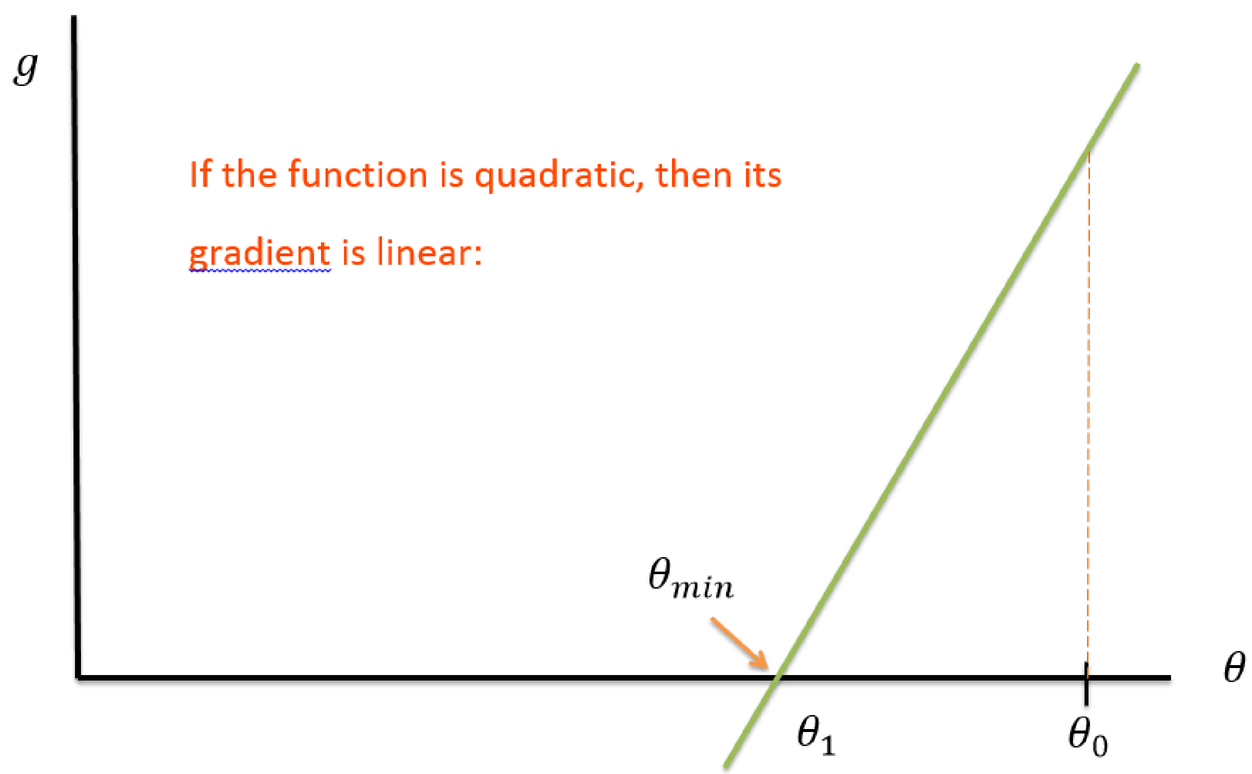
To find an extremum of f (.), solve $\frac{\partial f(\theta)}{\partial \theta} = g(\theta) = 0$.


g
θ_{min}
θ
θ_1
θ_0

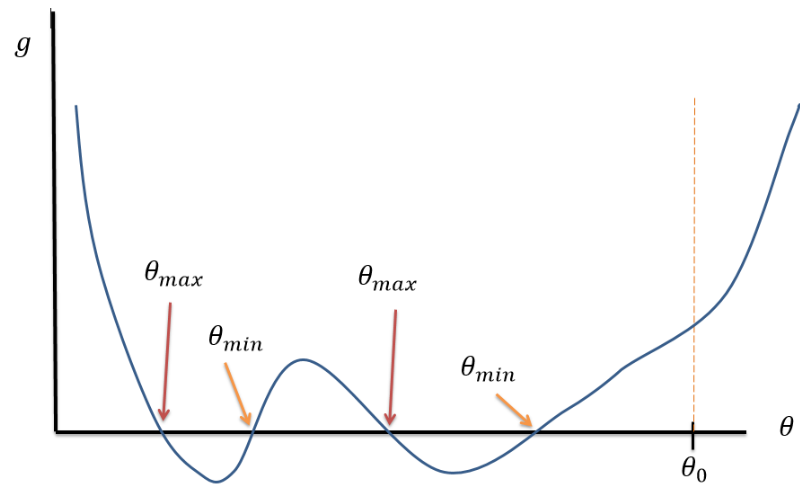

g
θ_{max}
θ_{min}
θ
θ_2
θ_1
θ_0

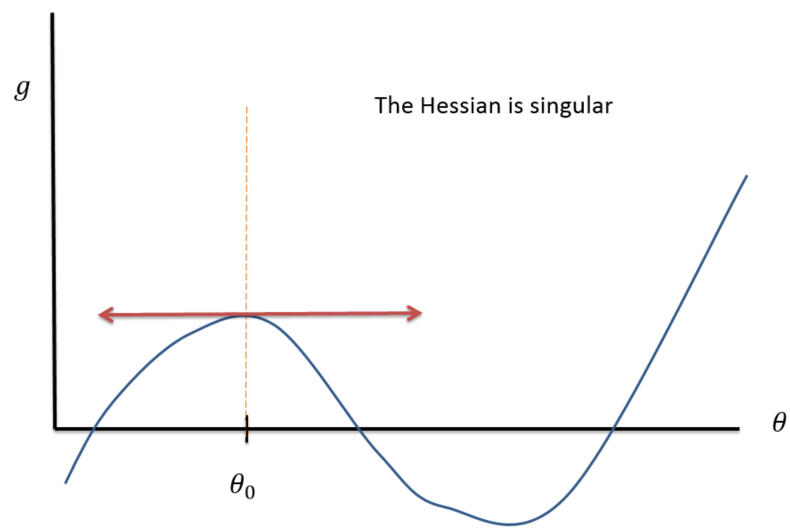


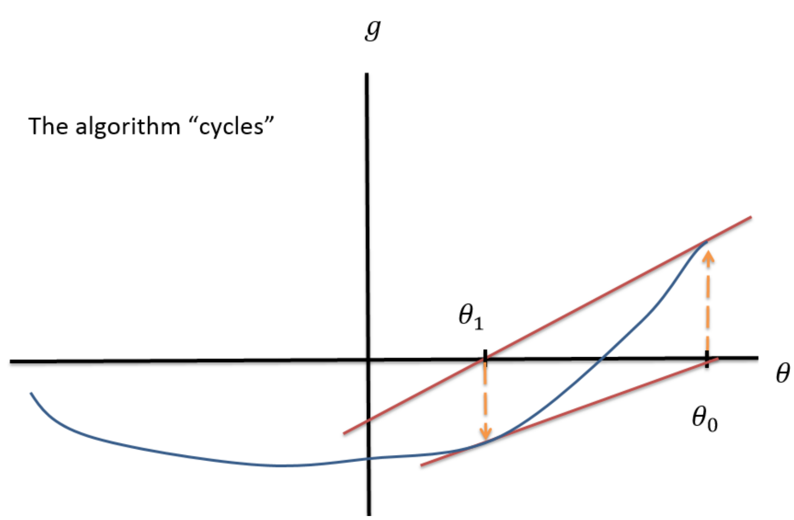
If $f(\boldsymbol{\theta})$ is *quadratic* in $\boldsymbol{\theta}$, then the algorithm converges in one iteration:



In general, different choices of θ_0 may lead to different solutions, or no solution at all.







Example (Where we actually know the answer)

$$f(\theta) = 3\theta^4 - 4\theta^3 + 1 \qquad \textit{locate minimum}$$

Analytically:

$$g(\theta) = 12\theta^3 - 12\theta^2 = 12\theta^2(\theta - 1)$$

$$H(\theta) = 36\theta^2 - 24\theta = 12\theta(3\theta - 2)$$

Turning points at $= 0,\ 0,\ 1$.

$$H(0) = 0 \qquad \textit{saddlepoint}$$

$$H(1) = 12 \qquad \textit{minimum}$$

Algorithm

$$\theta_{n+1} = \theta_n - H^{-1}(\theta_n)g(\theta_n)$$

$\theta_0 = 2$ (*say*)

$\theta_1 = 2 - \left(\frac{48}{96}\right) = 1.5$

$\theta_2 = 1.5 - \left(\frac{13.5}{45}\right) = 1.2$

$\theta_3 = 1.2 - \left(\frac{3.456}{23.040}\right) = 1.05$

$\vdots$

etc.

Try: $\theta_0 = -2;\quad \theta_0 = 0.5$