Newton's Method

This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dbricker@icaen.uiowa.edu

Reference: Section 3.5 of Linear & Nonlinear Programming, by A. Sofer and S. Nash

Solving Nonlinear Equations

Optimizing a Nonlinear Function
Solving Nonlinear Equations

- Single equation with single variable
  \[ g(x) = 0 \]

- System of \( n \) equations in \( n \) variables
  \[ g_i(x_1, x_2, \ldots x_n) = 0 \quad \forall i=1,2,\ldots n \]

Solving a Nonlinear Equation

\[ g(x) = 0 \]

...when an analytic solution doesn't exist.
For example,
\[
2x - \sin x = 0.5, \\
x^2 + x \log x = 1, \\
x^4 - 2x + 5 = 0, \text{ etc.}
\]

The Newton-Raphson method is an iterative, successive-approximation method for numerically solving such equations.
Let \( x_0 \) be an initial "guess" at the solution of
\[
g(x) = 0
\]
If \( g(x_0) \neq 0 \), then we wish to find a correction \( \delta \) so that
\[
g(x_0 + \delta) \approx 0
\]
Then \( x_1 = x_0 + \delta \) becomes our improved approximation, and we repeat the procedure until \( g(x_n) \) is "sufficiently close" to zero.

\[
\begin{align*}
\text{Solving a Nonlinear Equation} \\
g(x) &= 0
\end{align*}
\]

For some \( z \in [x^*, x] \),
\[
g(x) = g(x^*) + g'(x^*)(x - x^*) + \frac{g''(z)}{2} (x - x^*)^2
\]
Equivalently,
\[
\exists \alpha \in [0, 1] \text{ such that } g(x + \delta) = g(x) + g'(x)\delta + \frac{g''(x + \alpha \delta)}{2} \delta^2
\]
Suppose that \( x_k \) is a "guess" or approximation to the solution of \( g(x) = 0 \) at iteration \( k \).

\[
g(x_k + \delta) = g(x_k) + g'(x_k)\delta + \frac{g''(x_k + \alpha \delta)}{2} \delta^2 \approx 0
\]

Solve the equation

\[
g(x_k + \delta) = g(x_k) + g'(x_k)\delta = 0
\]

for the \( \delta \) which will give (hopefully) an improved approximate solution.

\[
\leftrightarrow \leftrightarrow
\]

\[
g(x_k) + g'(x_k)\delta = 0 \implies \delta = -\frac{g(x_k)}{g'(x_k)}
\]

The new (improved?) approximation is then

\[
x_{k+1} = x_k + \delta = x_k - \frac{g(x_k)}{g'(x_k)}
\]

\[
\leftrightarrow \leftrightarrow
\]
We are seeking the intersection of the graph of $g(x)$ and the $x$-axis.

$x_1 = x_0 + \delta$ is the intersection with the $x$-axis of the tangent line to the graph at $(x_0, g(x_0))$

Possible "Pathologies" of Newton's Method
Rate of Convergence

Define the error at iteration \( k \) by \( \epsilon_k = x_\bullet - x_k \)

Taylor's formula implies that, for some \( z \in [x_k, x_\bullet] \),

\[
0 = g(x_\bullet) = g(x_k + \epsilon_k) = g(x_k) + g'(x_k) \epsilon_k + \frac{1}{2} g''(z) \epsilon_k^2
\]

\[-\epsilon_k - \frac{g(x_k)}{g'(x_k)} = \frac{1}{2} \frac{g''(z)}{g'(x_k)} \epsilon_k^2
\]

\[-x_\bullet + x_k - \frac{g(x_k)}{g'(x_k)} = \frac{1}{2} \frac{g''(z)}{g'(x_k)} (x_\bullet - x_k)^2
\]

\(\Rightarrow\)

\[-x_\bullet + x_k - \frac{g(x_k)}{g'(x_k)} = \frac{1}{2} \frac{g''(z)}{g'(x_k)} (x_\bullet - x_k)^2
\]

\[x_{k+1} = \frac{1}{2} \frac{g''(z)}{g'(x_k)} (x_\bullet - x_k)^2
\]

\[\lim_{k \to \infty} \epsilon_k = 0 \Rightarrow x_{k+1} - x_\bullet \approx \frac{1}{2} \frac{g''(z)}{g'(x_k)} (x_\bullet - x_k)^2
\]

That is, the error is approximately squared at each iteration.

\[\Rightarrow\] The convergence is \textit{quadratic} with \( C = \left| \frac{1}{2} \frac{g''(z)}{g'(x_k)} \right| \)

\(\Rightarrow\)
\[ \forall Y \leftarrow X \quad \text{ROOT } C; \quad I; \quad \text{IO}; \quad \text{MAX\Delta IT} \]

1. Newton-Raphson method applied to finding a real root of a polynomial with coefficients C.
3. Author: Yuji Ijiri
4. \( i+1 \quad \diamond \quad \text{IO} \leftarrow 0 \quad \diamond \quad \text{MAX\Delta IT} \leftarrow 50 \)
5. \( \text{Next:} \quad X \leftarrow (Y+X) - X \cdot C \cdot X + X \cdot C \cdot \Phi \cdot o \cdot C \)
6. \( \rightarrow \text{Next IF } (Y \neq X) \land (\text{MAX\Delta IT} = i+1+1) \)

\[ 3X^3 + 2X^2 + X + 1 = 0 \]

<table>
<thead>
<tr>
<th>t</th>
<th>( X_t )</th>
<th>( G(X_t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000000000000000</td>
<td>7.0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0.5000000000000000</td>
<td>2.3750000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0.0476190476190476</td>
<td>1.0524781341107876</td>
</tr>
<tr>
<td>3</td>
<td>-0.8215623327982888</td>
<td>-0.1352096377777137</td>
</tr>
<tr>
<td>4</td>
<td>-0.7858722071987490</td>
<td>-0.0067344896901390</td>
</tr>
<tr>
<td>5</td>
<td>-0.7839000984497449</td>
<td>0.0000019706383372</td>
</tr>
<tr>
<td>6</td>
<td>-0.7838942937371222</td>
<td>0.0000000000170329</td>
</tr>
<tr>
<td>7</td>
<td>-0.7838942893686949</td>
<td>0.0000000000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t</th>
<th>( X_t )</th>
<th>( G(X_t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.0000000000000000</td>
<td>-1.0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>-0.8333333333333333</td>
<td>-0.1805555555555555</td>
</tr>
<tr>
<td>2</td>
<td>-0.7872340425531922</td>
<td>-0.0113943237933333</td>
</tr>
<tr>
<td>3</td>
<td>-0.7839108034766192</td>
<td>-0.0005604939292973</td>
</tr>
<tr>
<td>4</td>
<td>-0.7838942940928100</td>
<td>-0.0000000013778300</td>
</tr>
<tr>
<td>5</td>
<td>-0.7838942936869490</td>
<td>0.0000000000000000</td>
</tr>
</tbody>
</table>
\[ G(X) = 3X^3 + 2X^2 + X + 1 = 0 \]

<table>
<thead>
<tr>
<th>t</th>
<th>( X_t )</th>
<th>( G(X_t) )</th>
<th>( \varepsilon_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000000000000000</td>
<td>7.0000000000000000</td>
<td>-1.783894293686949</td>
</tr>
<tr>
<td>1</td>
<td>0.5000000000000000</td>
<td>2.3750000000000000</td>
<td>-1.283894293686949</td>
</tr>
<tr>
<td>2</td>
<td>0.0476190476190476</td>
<td>1.0524781341107877</td>
<td>-0.831513341305997</td>
</tr>
<tr>
<td>3</td>
<td>-0.8215623327982889</td>
<td>-0.1352096377771377</td>
<td>0.037668039111338</td>
</tr>
<tr>
<td>4</td>
<td>-0.7858722071987491</td>
<td>-0.0067344896901399</td>
<td>0.001977913511799</td>
</tr>
<tr>
<td>5</td>
<td>-0.7839009844974091</td>
<td>-0.0000197063833722</td>
<td>0.000005804762791</td>
</tr>
<tr>
<td>6</td>
<td>-0.7838942937371222</td>
<td>-0.0000000001703299</td>
<td>0.000000000050173</td>
</tr>
<tr>
<td>7</td>
<td>-0.7838942936869490</td>
<td>0.0000000000000000</td>
<td>0.0000000000000000</td>
</tr>
</tbody>
</table>

\[ \log_{10}(\text{Error vs Iteration \#}) \]
<table>
<thead>
<tr>
<th>k</th>
<th>$|\epsilon_{k+1}|/|\epsilon_k|$</th>
<th>$|\epsilon_{k+1}|/|\epsilon_k|^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.19714334100705E-1</td>
<td>4.03451222781368E-1</td>
</tr>
<tr>
<td>1</td>
<td>6.47849378453228E-1</td>
<td>5.04441355988410E-1</td>
</tr>
<tr>
<td>2</td>
<td>4.53005829734204E-2</td>
<td>5.44796826738463E-2</td>
</tr>
<tr>
<td>3</td>
<td>5.25090649383936E-2</td>
<td>1.39399517939303E0</td>
</tr>
<tr>
<td>4</td>
<td>2.93479100899355E-3</td>
<td>1.48376128317763E0</td>
</tr>
<tr>
<td>5</td>
<td>8.64345378706824E-6</td>
<td>1.48902790661432E0</td>
</tr>
<tr>
<td>6</td>
<td>2.21278102318995E-6</td>
<td>4.41028491391872E4</td>
</tr>
</tbody>
</table>
\[ \frac{\| \varepsilon_{k+1} \|}{\| \varepsilon_k \|^2} \]

\[ G(X) = 3X^3 + 2X^2 + X + 1 \]

There appears to be only one real root!
\[ G(X) = X^3 - 2X^2 - X + 2 \]

Roots are \(-1, 1, \& 2\)

<table>
<thead>
<tr>
<th>( t )</th>
<th>( X_t )</th>
<th>( G(X_t) )</th>
<th>( \varepsilon_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.50000000000000</td>
<td>1.87500000000000</td>
<td>-0.50000000000000</td>
</tr>
<tr>
<td>1</td>
<td>-1.57142857142857</td>
<td>-5.24781341107708</td>
<td>0.57142857142857</td>
</tr>
<tr>
<td>2</td>
<td>-1.15801561782269</td>
<td>-1.07688366081184</td>
<td>0.15801561782269</td>
</tr>
<tr>
<td>3</td>
<td>-1.01733958834521</td>
<td>-0.10546050034808</td>
<td>0.10733958834521</td>
</tr>
<tr>
<td>4</td>
<td>-1.00024516687396</td>
<td>-0.00147130179250</td>
<td>0.00024516687396</td>
</tr>
<tr>
<td>5</td>
<td>-1.00000000000000</td>
<td>-0.00000000000000</td>
<td>0.00000000000000</td>
</tr>
<tr>
<td>6</td>
<td>-1.00000000000000</td>
<td>-0.00000000000000</td>
<td>0.00000000000000</td>
</tr>
<tr>
<td>7</td>
<td>-1.00000000000000</td>
<td>0.00000000000000</td>
<td>0.00000000000000</td>
</tr>
</tbody>
</table>
Log (base 10) of Error vs Iteration #

<table>
<thead>
<tr>
<th>k</th>
<th>$\frac{|e_{k+1}|}{|e_k|}$</th>
<th>$\frac{|e_{k+1}|}{|e_k|^2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$1.14285714285714E0$</td>
<td>$2.28571428571429E0$</td>
</tr>
<tr>
<td>1</td>
<td>$2.7652731189711E-1$</td>
<td>$4.83922829581994E-1$</td>
</tr>
<tr>
<td>2</td>
<td>$1.09733383219595E-1$</td>
<td>$6.94466439735635E-1$</td>
</tr>
<tr>
<td>3</td>
<td>$1.413914039274E-2$</td>
<td>$8.15425378690662E-1$</td>
</tr>
<tr>
<td>4</td>
<td>$2.04242302038894E-4$</td>
<td>$8.33074626826414E-1$</td>
</tr>
<tr>
<td>5</td>
<td>$4.43437828748177E-9$</td>
<td>$8.85574806157926E-1$</td>
</tr>
</tbody>
</table>
Does not appear to be converging to a positive $C$

Appears to be converging to a $C = \text{approximately 1}$
Secant Method

Often the derivative \( g'(x) \) is difficult to compute, making the Newton-Raphson method undesirable. The secant method avoids the use of derivatives by finding the intersection with the \( x \)-axis not of the tangent line, but a secant line.

Two initial "guesses" are required.

System of \( n \) equations in \( n \) variables

Consider the system of nonlinear equations

\[
\begin{align*}
g_1(x_1, x_2, \ldots, x_n) &= 0 \\
g_2(x_1, x_2, \ldots, x_n) &= 0 \\
&\vdots \\
g_n(x_1, x_2, \ldots, x_n) &= 0
\end{align*}
\]

Dropping the quadratic terms from Taylor's Formula:

\[
g_i(x^i + \Delta^i) \approx g_i(x^i) + \nabla g_i(x^i) \Delta^i \quad \forall \quad i = 1, 2, \ldots, n
\]
If we choose the step $\Delta^t$ so that $g_i(x^t + \Delta^t) \approx 0$ we get the system of linear equations

$$\nabla g_i(x^t) \Delta^t = -g_i(x^t) \quad \forall i = 1, 2, \ldots n$$

which we must solve for $\Delta^t$

Let $J(x_1, x_2, \ldots x_n)$ be the Jacobian of this system, evaluated at $x$, i.e., the $n \times n$ matrix with row $i$ equal to $\nabla g_i(x_1, x_2, \ldots x_n)$, the gradient of the $i^{th}$ constraint function.

The coefficient matrix of this linear system is the Jacobian Matrix

$$J(x^t) = \begin{bmatrix}
\frac{\partial g_1(x^t)}{\partial x_1} & \frac{\partial g_1(x^t)}{\partial x_2} & \ldots & \frac{\partial g_1(x^t)}{\partial x_n} \\
\frac{\partial g_2(x^t)}{\partial x_1} & \frac{\partial g_2(x^t)}{\partial x_2} & \ldots & \frac{\partial g_2(x^t)}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial g_n(x^t)}{\partial x_1} & \frac{\partial g_n(x^t)}{\partial x_2} & \ldots & \frac{\partial g_n(x^t)}{\partial x_n}
\end{bmatrix}$$
The **Newton-Raphson Method** for solving
\[\begin{align*}
g_1(x_1, x_2, \ldots, x_n) &= 0 \\
g_2(x_1, x_2, \ldots, x_n) &= 0 \\
&\vdots \\
g_n(x_1, x_2, \ldots, x_n) &= 0
\end{align*}\]

Let \(x^t\) be the approximation to the solution at iteration \(\#t\). Then the next approximation is \(x^{t+1} = x^t + \Delta^t\) where \(\Delta^t = -[J(x^t)]^{-1}g(x^t)\)

\[\text{inverse of the Jacobian matrix}\]

\[\leftarrow \rightarrow\]

\[\forall y \leftarrow \text{NEWTON\_RAPHSON} \ x; y; j; dy\]

1. Solve the nonlinear system of equations \(G(x) = 0\), where \(G\) is a vector-valued function.
2. \(G\) and Jacobian must be user-defined, as well as TOL and MAX\_ITER.
3. UTCNL, 'Tolerance \(\text{on} \ / \ |G(x)| = \text{',',\text{TOL}\text{)}\)
4. start\_timer = ITER\(\text{+1} \odot Y \leftarrow X \odot UTCNL\)
5. \(XDPATH\leftarrow (\rho X), 1 \odot \rho X\)
6. NEXT; \(V \leftarrow G \ Y\)
7. \(\rightarrow\text{STOP IF TOL} \geq f / |V|\)
8. \(J \leftarrow \text{JACOBIAN} \ Y\)
9. \(DY \leftarrow -V \cdot J\)
10. \(\rightarrow\text{bypass IF } \sim \text{ detail } \odot \text{NR\_OUTPUT}\)
11. \(\text{bypass; } XDPATH \leftarrow XDPATH, Y + Y + DY\)
12. \(\rightarrow\text{NEXT IF MAX\_ITER} \geq \text{ITER} \odot \text{ITER} + 1\)
13. 'Warning... Newton-Raphson algorithm did not converge.'
14. \(\rightarrow 0\)
15. \(\text{STOP; UTCNL, '*** Converged ***'}\)
16. '(Max. Abs. Value \leq \text{tolerance, ',',(|TOL),')}\)
EXAMPLE

Solve:
\[
\begin{align*}
\frac{3000}{x_1x_2} &= 1 \\
\frac{4000}{x_1^2x_2} &= 2
\end{align*}
\]

i.e.,
\[
\begin{align*}
g_1(x_1,x_2) &= 1 - 3000x_1x_2^2 = 0 \\
g_2(x_1,x_2) &= 2 - 4000x_1^2x_2 = 0
\end{align*}
\]

\[\vartheta \leftarrow \text{JACOBIAN } \vartheta \]

\[\begin{align*}
[1] & \quad \vartheta \\
[2] & \quad \text{EXAMPLE EQUATIONS FOR NEWTON-RAPHSON ALGORITHM} \\
[3] & \quad \text{JACOBIAN MATRIX FOR SAMPLE SYSTEM OF EQUATIONS TO BE SOLVED BY NEWTON-RAPHSON ALGORITHM}
\end{align*}\]
We will start at \( x = (x_1, x_2) = (15, 15) \)

and terminate when

\[ |g_1(x)| \leq 10^{-10} \quad \text{and} \quad |g_2(x)| \leq 10^{-10} \]

Tolerance (on \( \|G(x)\| \)) = 0.0000000001

**Iteration 1**

\( x = 15 \ 15 \)

Function Values G(X) = 0.111111 0.814815

Jacobian Matrix:

\[
\begin{bmatrix}
0.118519 & 0.0592593 \\
0.0790123 & 0.158025
\end{bmatrix}
\]

(Determinant = 0.0140468)

Step is 2.1875 -6.25

with length 6.62178
Iteration 2

\[ X = 17.1875 \ 8.75 \]

Function Values \( g(X) = -0.160614 \ -1.0397 \)

Jacobian Matrix:

\[
\begin{bmatrix}
0.135053 & 0.132642 \\
0.176855 & 0.694789 \\
\end{bmatrix}
\]

(Determinant = 0.0703752)

Step is \(-0.373925 \ 1.59161\)

with length 1.63494
Iteration 3

\[ \begin{align*}
X &= 16.8136 \ 10.3416 \\
\text{Function Values } G(X) &= -0.026155 \ -0.224455 \\
\text{Jacobian Matrix:} &
\begin{pmatrix}
0.122063 & 0.0992258 \\
0.132301 & 0.430195 \\
\end{pmatrix}
\end{align*} \]

(Determinant = 0.0393831)

Step is \(-0.279815 \ 0.607805\)
with length 0.869121

Iteration 4

\[ \begin{align*}
X &= 16.5338 \ 10.9494 \\
\text{Function Values } G(X) &= -0.00227541 \ -0.0179319 \\
\text{Jacobian Matrix:} &
\begin{pmatrix}
0.12124 & 0.0915369 \\
0.122049 & 0.368592 \\
\end{pmatrix}
\end{align*} \]

(Determinant = 0.033516)

Step is \(-0.0239508 \ 0.0565805\)
with length 0.061441
Iteration 5

\[ X = 16.5098 \quad 11.006 \]
Function Values \( G(X) = -0.000017987 \quad -0.000134764 \)
Jacobian Matrix:
\[
\begin{bmatrix}
0.121142 & 0.0908612 \\
0.121148 & 0.363463 \\
\end{bmatrix}
\]
(Determinant = 0.033023)

Step is \(-0.000172825 \quad 0.000428384\)
with length 0.000461932

Iteration 6

\[ X = 16.5096 \quad 11.0064 \]
Function Values \( G(X) = -1.02881E^{-9} \quad 7.67939E^{-9} \)
Jacobian Matrix:
\[
\begin{bmatrix}
0.121141 & 0.090856 \\
0.121141 & 0.363424 \\
\end{bmatrix}
\]
(Determinant = 0.0330193)

Step is \(-9.80709E^{-9} \quad 2.43997E^{-8}\)
with length 2.62968E^{-8}
*** Converged ***
(Max. Abs. Value ≤ tolerance, 0.0000000001)

Final Solution

X = 16.5096 11.0064
G(X) = 3.33067E-16 0
CPU time: 11.95 sec.
# of iterations = 7
Secant Method

Optimizing a Nonlinear Function

Function of a single variable
Minimize $f(x)$

Function of several variables
Minimize $f(x_1, x_2, \ldots, x_n)$
Function of a single variable
Minimize \( f(x) \)

Consider the problem of minimizing a function of several variables:

Minimize \( f(x_1, x_2, \ldots, x_n) \)

Suppose that \( f \) is differentiable, i.e., the gradient

\[
\nabla f(x) = \begin{bmatrix}
\frac{af}{\partial x_1} \\
\frac{af}{\partial x_2} \\
\vdots \\
\frac{af}{\partial x_n}
\end{bmatrix}
\]

is defined.

Necessary Condition for Optimality:
If \( X^* \) is optimal, then
\[
\nabla f(X^*) = 0, \text{ i.e., } \\
\frac{af(X^*)}{\partial x_i} = 0 \text{ for each } i = 1, 2, \ldots, n
\]
This necessary condition for optimality yields a system of (in general) $n$ **nonlinear** equations in $n$ variables.

**Newton's Method** is simply the application of the Newton-Raphson method to solving this system of equations.

Applying the Newton-Raphson method to the solution of the equations $\nabla f(x^*) = 0$, i.e.,

\[
\begin{align*}
    g_1(x_1, x_2, \ldots, x_n) &= \frac{af}{ax_1} = 0 \\
    g_2(x_1, x_2, \ldots, x_n) &= \frac{af}{ax_2} = 0 \\
    \vdots \\
    g_n(x_1, x_2, \ldots, x_n) &= \frac{af}{ax_n} = 0
\end{align*}
\]

we find that the Jacobian matrix of this system is the **Hessian** matrix of the function $f$, i.e., $\nabla^2 f(x)$.
Hessian Matrix

\[ \nabla^2 f(x^t) \equiv \begin{bmatrix} \frac{\partial^2 f(x^t)}{\partial x_1^2} & \frac{\partial^2 f(x^t)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x^t)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x^t)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x^t)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x^t)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x^t)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x^t)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x^t)}{\partial x_n^2} \end{bmatrix} \]

\[ \leftrightarrow \leftrightarrow \]

Newton's Method

Let \( x^t \) be the approximation to the solution at iteration \( t \).

Then the approximation at iteration \( t+1 \) is given by

\[ x^{t+1} = x^t + \Delta^t, \text{ where } \Delta^t \text{ is the solution of the (linear) equation} \]

\[ \nabla^2 f(x^t) \Delta^t = -\nabla f(x^t) \]

i.e.,

\[ \Delta^t = -[\nabla^2 f(x^t)]^{-1} \nabla f(x^t) \]

\[ \leftrightarrow \leftrightarrow \]
**Comments**

- If \( f(x) \) is a quadratic function, Newton's method converges to a stationary point in a single iteration.
- Newton's method does not discriminate among minimizers, maximizers, and saddle points.
- If the Hessian matrix at an iteration is positive definite, then the direction \( s_t \) is a direction of descent, i.e., \( f(x_t + \epsilon \Delta_t) < f(x_t) \) for some \( \epsilon > 0 \), even though it might be that \( f(x_t + \Delta_t) \geq f(x_t) \).
- It is more efficient and more numerically stable to solve the system of equations \( \nabla^2 f(x_t) \Delta_t = -\nabla f(x_t) \)

by means other than inverting the Hessian matrix.

---

**Example**

Minimize \( f(x_1, x_2) = (x_2 - x_1^2)^2 + (1 - x_1)^2 \)

\[
\nabla f(x) = \begin{bmatrix}
\frac{\partial f}{\partial x_1} \\
\frac{\partial f}{\partial x_2}
\end{bmatrix}
= \begin{bmatrix}
4x_1^3 - 4x_1x_2 + 2x_1 - 2 \\
2(x_2 - x_1)
\end{bmatrix}
\]

\[
\nabla^2 f(x) = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\
\frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2}
\end{bmatrix}
= \begin{bmatrix}
12x_1^2 - 4x_2 + 2 & -4x_1 \\
-4x_1 & 2
\end{bmatrix}
\]
Let's begin at the point \( x^0 = (2, 1) \)

\[
\begin{align*}
\Delta x & = 2 \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\
\Gamma(x) & = 10 \\
\nabla \Gamma(x) & = 26 \begin{bmatrix} -6 \end{bmatrix} \\
\text{Hessian Matrix} & = \begin{bmatrix} 46 & -8 \\ -8 & 2 \end{bmatrix}
\end{align*}
\]

We need to solve the equations
\[
\begin{cases}
46 \delta_1 - 8 \delta_2 = -26 \\
-8 \delta_1 + 2 \delta_2 = 6
\end{cases}
\]

\[
\Rightarrow \begin{align*}
\delta_1 & = -0.1428 \\
\delta_2 & = 2.4286
\end{align*}
\]

\[
\begin{align*}
x_1 & = x_1^0 + \delta_1 = 2 - 0.1428 \\
x_2 & = x_2^0 + \delta_2 = 2 + 2.4286
\end{align*}
\]

\[
\begin{align*}
x & = 1.857142857 \begin{bmatrix} 3.428571429 \\ 0.7351103707 \end{bmatrix} \\
\Gamma(x) & = 1.885889213 \begin{bmatrix} -0.04091632653 \end{bmatrix} \\
\nabla \Gamma(x) & = 1.885889213 \begin{bmatrix} -0.04091632653 \end{bmatrix} \\
\text{Hessian Matrix} & = \begin{bmatrix} 29.67346939 & -7.428571429 \\ -7.428571429 & 2 \end{bmatrix}
\end{align*}
\]

Improvement: 9.264889629
Iteration 3

\[ x = 1.033513445 \quad 0.3901560624 \]
\[ f(x) = 0.4610800424 \]
\[ \nabla f(x) = 2.871218307 \quad -1.356401384 \]

Hessian Matrix =

\[
\begin{bmatrix}
13.2596568 & -4.134453782 \\
-4.134453782 & 2
\end{bmatrix}
\]

Improvement: 0.2740243283

Iteration 4

\[ x = 1.019348709 \quad 1.039983907 \]
\[ f(x) = 0.0003744193915 \]
\[ \nabla f(x) = 0.03952709666 \quad -0.000405965428 \]

Hessian Matrix =

\[
\begin{bmatrix}
10.31338825 & -4.077948395 \\
-4.077948395 & 2
\end{bmatrix}
\]

Improvement: 0.4607116285
Iteration 5

\[ x = 1.000007871 \ 0.9996416742 \]
\[ \Gamma(x) = 1.399888212E^{-7} \]
\[ \nabla \Gamma(x) = 0.001512025877 \ -0.0007481359977 \]

Hessian Matrix =

\[
\begin{bmatrix}
10.00152221 & -4.000031484 \\
-4.000031484 & 2
\end{bmatrix}
\]

Improvement: 0.0003742739427

Iteration 6

\[ x = 1.000000006 \ 1.000000012 \]
\[ \Gamma(x) = 3.462781953E^{-17} \]
\[ \nabla \Gamma(x) = 1.201587541E^{-8} \ -1.237219216E^{-10} \]

Hessian Matrix =

\[
\begin{bmatrix}
10.00000009 & -4.000000024 \\
-4.000000024 & 2
\end{bmatrix}
\]

Improvement: 1.399888212E^{-7}
Convergence criterion satisfied:

Improvement in objective
Step size
Gradient
*** CONVERGED ***

Solution found is 1.00000000600000012
where F is 3.462781953E-17
and VF is 1.201587541E-8 -1.237219216E-10

# iterations = 6
Elapsed CPU time: 11.33 seconds
Disadvantages

- Newton’s method converges to a stationary point, which may or may not be a minimizer.

- Newton’s method requires the computation of $\frac{1}{2} n^2$ 2nd partial derivatives (i.e., the Hessian matrix).

- Newton’s method requires at each iteration the inversion of a matrix of order $n$ (or the solution of a linear system of equations in $n$ variables).

For these reasons, Newton’s method is not of practical significance for unconstrained optimization.