§3. Newton's method for coupled nonlinear equations

(a) Newton-Raphson for 1 equation:
\[ f(x) = 0 = f(x_i + \delta_i) \]
\[ = f(x_i) + f'(x_i) \cdot \delta_i + \cdots \]
\[ \delta_i = - \frac{f(x_i)}{f'(x_i)} \]
\[ x_{i+1} = x_i + \delta_i = x_i - \frac{f(x_i)}{f'(x_i)} \]

(b) Newton-Raphson for 2 equations:
\[ f_1(x, y) = 0 \]
\[ f_2(x, y) = 0 \]
Guess \((x_0, y_0)\). Let
\[ \begin{align*}
  f_1(x_0 + \delta x, y_0 + \delta y) &= 0 \\
  f_2(x_0 + \delta x, y_0 + \delta y) &= 0 \\
  \Rightarrow \quad f_1(x_0, y_0) + \frac{\partial f_1}{\partial x} \delta x + \frac{\partial f_1}{\partial y} \delta y &= 0 \\
  f_2(x_0, y_0) + \frac{\partial f_2}{\partial x} \delta x + \frac{\partial f_2}{\partial y} \delta y &= 0
\end{align*} \]

\( \times \) Nonlinear system \( \rightarrow \) Iterating linear systems. Some
\( \times \) Nonlinear system \( \rightarrow \) Iterating linear systems. Some
\( \times \) Nonlinear system \( \rightarrow \) Iterating linear systems. Some

Jacobian:
\[ \left( \begin{array}{c} \delta x \\ \delta y \end{array} \right) = - \begin{pmatrix} f_1(x_0, y_0) \\ f_2(x_0, y_0) \end{pmatrix} \]

(c) Newton-Raphson for \( n \) equations:
\[ f_i(x_1, \ldots, x_n) = 0, \quad i = 1, n. \]
\[
\begin{pmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n}
\end{pmatrix}
\begin{pmatrix}
\delta x_1 \\
\delta x_2 \\
\vdots \\
\delta x_n
\end{pmatrix}
= 
\begin{pmatrix}
f_1 \\
f_2 \\
\vdots \\
f_n
\end{pmatrix}
\]

Noting the \(n \times n\) matrix is the Jacobian, we write
\[
J \cdot \delta x = -f \quad \Rightarrow \quad \delta x = -J^{-1}f
\]
\[
\Rightarrow \quad x^{(n+1)} = x^{(n)} - J^{-1}f(x^{(n)})
\]

Remarks:

(i) Numerical evaluation of \(J\): What if \(f\) not given as simple analytical expression? Will see such situations in, say, shooting method.

\[
\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(x_j + \delta) - f_i(x_j)}{\delta}
\]

(ii) \(J\) doesn't have to be very accurate initially. "Delayed updating of \(J\)" saves efforts.

(iii) How to solve the linear system \(J \cdot \delta x = -f\), or how to invert \(J^{-1}\)? Any linear solver. Could do LU decomposition, and use the same \(J^{-1}\) for multiple iterations.

(iv) How to facilitate convergence of Newton's iteration?

- Start close to solution; the value of a good initial guess. Use physical intuition or constraint estimation.

\[
x^{(n+1)} = x^{(n)} + \omega \cdot \delta x^{(n)}, \quad \omega < 1
\]

(Recall graphical argument)
(a) Globally convergent Newton's scheme

(See Numerical Recipes, §9.7).

When close to the root, Newton's method converged quickly. But the key is to choose the initial guess judiciously. See figure.

Is there a way to guard against this type of "runaway" situation? How about not taking the full Newton step?

(i) descent direction: To find the root for \( f(x) = 0 \), we guess \( x_{00} \) and want \( f(x_{00} + \delta x) = 0 \)

\[ \Rightarrow \delta x = -J^{-1}f(x_{00}) \]

Note that to go along the descent of \( \delta x \), \( f \) is supposed to decrease in magnitude. To see this, write \( \nabla f = \frac{1}{2} f \cdot J \) so \( \nabla f = f \cdot J \)

\[ \Rightarrow \nabla f \cdot \delta x = (f \cdot J) \cdot (-J^{-1}f) = -f^T f < 0 \]

Thus, \( \delta x \) gives a descending direction for \( |f| \), at least if one takes very small steps.

(ii) Backtracking: Let's take a full Newton step \( \delta x \) and compare \( f(x_{00}) \) and \( f(x_{\text{new}}) \).

If \( f(x_{\text{new}}) \leq f(x_{00}) \), we have stepped too far.

We backtrack and try

\[ x_{\text{new}} = x_{00} + \lambda \cdot \delta x, \ 0 < \lambda < 1 \]

and see if that reduces \( f \) or \( |f| \). This will lead to a "globally convergent" method.

* Optimum \( \lambda \)? See NR §9.7 (pp 376-381), with code.
* How to minimize \( F(\tilde{x}_{\text{new}}) = F(\tilde{x}_{\text{old}} + \lambda \delta x) \)?

Write \( G(\lambda) = F(\tilde{x}_{\text{new}}) = F(\tilde{x}_{\text{old}} + \lambda \delta x) \). We seek to minimize \( G(\lambda) \) with respect to \( \lambda \). But directly taking \( G'(\lambda) = 0 \) won't get us a solution, since

\[
G'(\lambda) = \nabla F(\lambda) \cdot \delta x = f(\lambda) \cdot \nabla x \cdot \delta x = 0
\]

is harder to solve than \( f(\lambda) = 0 \). So we try to "model" \( G(\lambda) \).

* We know \( G(0) = F(\tilde{x}_{\text{old}}) \), \( G(1) = F(\tilde{x}_{\text{new}}) \) and also \( G'(0) = \nabla F(\tilde{x}_{\text{old}}) \cdot \delta x = -f(\tilde{x}_{\text{old}}) \cdot \delta x \).

We model \( G(\lambda) \) as 2nd order polynomial:

\[
G(\lambda) = G(0) + G'(0) \lambda + p \lambda^2
\]

\[\Rightarrow G(1) = G(0) + G'(0) + p \Rightarrow p = G(0) - G(0) - G'(0) \]

Now \( G'(0) = 0 = G'(0) + 2p \lambda \Rightarrow \lambda = \frac{1}{2} \frac{-G'(0)}{[G(1) - G(0)]}/G'(0) \)

Note that \( \lambda > 0 \), \( \lambda < \frac{1}{2} \), since \( G(0) < 0 \), \( G(1) > G(0) \). \[\Rightarrow \text{This gets us } \lambda_1 = \lambda. \text{ Take } \tilde{x}_{\text{new}} = \tilde{x}_{\text{old}} + \lambda_1 \cdot \delta x \]

* What if \( \tilde{x}_{\text{new}} \) is no good? Try another backtrack using \( G(0), G'(0), G(1), G(\lambda_1) \) in 3rd order model:

\[
G(\lambda) = G(0) + G'(0) \lambda + b \lambda^2 + a \lambda^3
\]

\[
\begin{align*}
G(0) &= G(0) + G'(0) + b + a \\
G(1) &= G(0) + G'(0) \lambda_1 + b \lambda_1^2 + a \lambda_1^3
\end{align*}
\]

\[\Rightarrow \text{minimize this } G(\lambda) \text{ to get } \lambda_2. \]

* If necessary use \( G(0), G'(0), G(\lambda_1), G(\lambda_2) \) to get \( \lambda_3 \), etc.