# The Newton-Raphson Method and its Application to Fixed Points

*Jonathan Tesch, 21 Nov. 2005*

## 1. The Newton-Raphson Algorithm

The Newton-Raphson algorithm is a numerical method for finding the roots of a function. It does so by computing the Jacobian linearization of the function around an initial guess point, and using this linearization to move closer to the nearest zero.

Consider a function $H:\mathbf{R}^n \rightarrow \mathbf{R}^n$ that has a zero at $x^*$, i.e. $H(x^*)=0$. If we set $y=H(x)$, we can approximate changes in $y$, $\Delta y$, due to changes in $x$, $\Delta x$, by linearizing $H$ around some point $x^i$.

$$\Delta y = J_H(x^i)\Delta x \tag{1}$$

where $J_H(x^i)$ is the Jacobian of $H$ at $x^*$. We seek a zero of $H$, so we set $\Delta y=-H(x^i)$ and solve (1) for $\Delta x$ to obtain

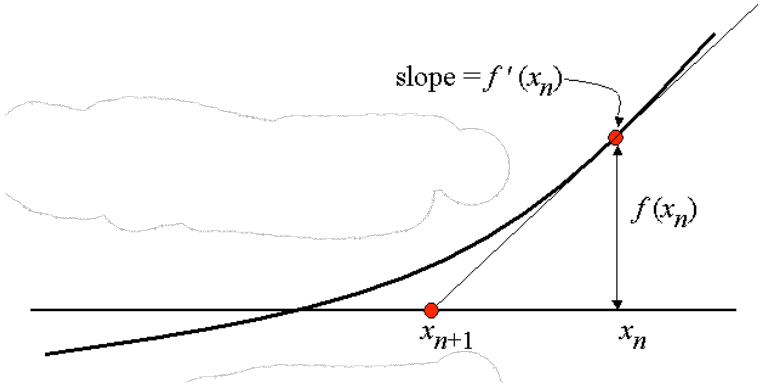
$$\Delta x = -J_H(x^i)^{-1} H(x^i) \tag{2}$$

Because (1) is a Jacobian linearization of $H$, (2) provides a change in $x$ that that moves closer to the desired zero. A second iteration of this process can now be performed at a point $x^{i+1}$ such that $x^{i+1}= x^i+\Delta x$. Using (2), this becomes

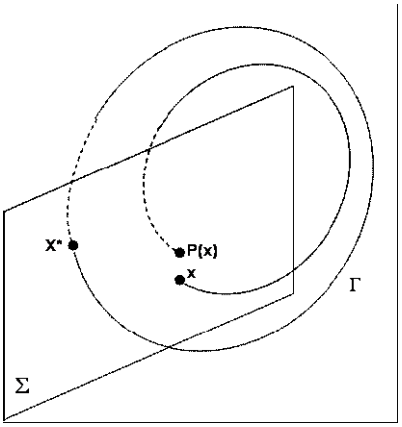
$$x^{i+1} = x^i - J_H(x^i)^{-1} H(x^i) \tag{3}$$

(3) can be used iteratively from some initial guess to yield a better and better approximation of $x^*$. The algorithm terminates once a value of $x$ is reached such that $H(x)$ is sufficiently close to zero.

The image below shows a graphical representation of one iteration of the method for a function $f:\mathbf{R}\rightarrow\mathbf{R}$.



## 2. Newton-Raphson and Fixed Points




**Poincaré plane Σ intersecting a state trajectory.**

The Newton-Raphson method can be used to find the fixed points of systems that have stable limit cycles. Consider a hybrid system $H$ with guard condition $G \in$ TQ, where TQ is the tangent bundle of the configuration space of $H$. To find a fixed point of the system, we construct a Poincaré hyperplane Σ such that

$$\Sigma = \left\{(q,\dot{q}) \in TQ | G\right\}$$

In essence, this plane samples the phase trajectory of the system each time the guard condition is reached. We now define a mapping $P: \Sigma \rightarrow \Sigma$, i.e. P is a mapping from a state on the Poincaré plane back to the plane: $P = x_{k+1} = \phi(x_k)$, where $\phi$ is the flow of the state trajectory. Computationally, $\phi$ is simply the forward integration of the hybrid system's underlying equations of motion, run until the guard condition is met.

A fixed point is defined as a location on the Poincaré plane, $x^*$, such that $x^* = \phi(x^*)$. We can find this value from some initial guess point $x^i$ by establishing an error function $E(x) = \phi(x) - x$ that provides the difference between a state vector and the result of the forward integration from that point along the phase trajectory. This error is zero for a fixed point, thus finding a fixed point corresponds to finding the zero of $E$, which can be accomplished using the multidimensional form of the Newton-Raphson algorithm.

If we start from an initial guess for the fixed point, $x^i$, then (3) becomes

$$x^{i+1} = x^i - J_E(x^i)^{-1} E(x^i) \tag{4}$$

The task now is to determine the Jacobian matrix of the error function at the point $x^i$. If the phase space is a subset of $\mathbf{R}^n$, then the error function $E$ is a mapping $E: \mathbf{R}^n \rightarrow \mathbf{R}^n$, where the ith element in $E$, $E_i$, defines the error of the ith state variable. The Jacobian for such a function is an $nxn$ matrix of the form

$$J_E(x_1,...,x_n) = \begin{bmatrix} \frac{\partial E_1}{\partial x_1} & \cdots & \frac{\partial E_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial E_n}{\partial x_1} & \cdots & \frac{\partial E_n}{\partial x_n} \end{bmatrix} \tag{5}$$

Note that each column contains the derivatives of $E$ with respect to a particular state variable. The derivatives contained in (5) can be computed by perturbing each state variable in $x^i$ in turn by a small amount, Δ, along the Poncaré plane, and evaluating the resulting vector with the flow $\phi$. We then apply the error function to the result. When Δ is small, this process estimates the derivative of the error with respect to the perturbed variable: $\partial E / \partial x_n$. For instance, to compute the first column of (5), we perturb the first state variable in our initial guess by a small amount, while keeping the other variables the same:

$$x^i + dx_1 = \begin{bmatrix} x_1^i + \Delta_1 \\ x_2^i \\ \vdots \\ x_n^i \end{bmatrix}$$

We now evaluate the error function with this perturbed variable, giving us the change in *E* due to the small change in $x^i$

$$\Delta E = E(x^i + dx_1) - E(x^i) = \left[\phi(x^i + dx_1) - (x^i + dx_1)\right] - \left[\phi(x^i) - x^i\right] \quad (6)$$

When $\Delta_1$ becomes small, $\Delta E \to \partial E$. Dividing by $\Delta_1$ allows us to approximate the derivative

$$\frac{\Delta E}{\Delta_1} \approx \frac{\partial E}{\partial x_i}$$

which corresponds to the first column of the Jacobian matrix of *E*. This process can be repeated for each state variable in $x^i$, eventually yielding the matrix in (5).

With the Jacobian identified, (4) can be used to find the next guess for the fixed point. The loop terminates once all the elements in the error function are below some small number $\varepsilon>0$ sufficiently close to zero, i.e. when

$$\left|\max\left\{E(x^i)\right\}\right| < \varepsilon$$

Note that convergence of (4) does not occur if at any time *i,* the error increases with the next iteration, *i+1*.