Problem (1) Finding roots of a function: find $x^* \in R^n$ such that $f(x^*)=0$
Newton's Algorithm without a Linesearch:
Step 1. Given an initial point x_0; iteration = 0
Step 2. For k=0,1,2,….until convergence; if iter ≤ max_iter
Step 3. Newton's Direction: $f'(x_k)\Delta x=-f(x_k)$; $J(x_k)\Delta x=-F(x_k)$ where J(x) is the Jacobian
Step 4. Update: $x_{k+1}=x_k+\Delta x$

Problem (2) Find the minimum value: find $x^* \in R^n$ such that $f'(x^*)=0$
Newton's Algorithm without a Linesearch:
Step 1. Given an initial point x_0; iteration = 0
Step 2. For k=0,1,2,….until convergence; if iter ≤ max_iter
Step 3. Newton's Direction: $f''(x_k)\Delta x=-f'(x_k)$; $H(x_k)\Delta x=-J(x_k)$; J(x)→Jacobian, H(x)→Hessian
Step 4. Update: $x_{k+1}=x_k+\Delta x$

A simple parallelization of the serial algorithms shown above is to include multiple initial points in Step 1. Then, assign a processor to run the serial Newton calculation for each different initial point in parallel. After the calculation is completed by each worker processor, the results can be sent to the master node to determine the result which most closely meets the set convergence criterion.

The Newton method without a Linesearch is considered to be a local method. Therefore, the convergence of the method depends highly on the quality of the initial starting point. The use of multiple starting points alleviates the difficulties associated with a local method. By feeding the parallel calculation various starting points, it is more likely to obtain a converged calculation(s) which offsets any calculations which did not meet the convergence criterion.