Nonlinear equations

Introduction

Non-linear equations or *root-finding* is a problem of finding a set of n variables $\{x_1, \ldots, x_n\}$ which satisfy n equations

$$f_i(x_1, ..., x_n) = 0 \ , \ i = 1, \ldots, n \ , \tag{1}$$

where the functions f_i are generally non-linear.

Newton's method

Newton's method (also reffered to as Newton-Raphson method, after Isaac Newton and Joseph Raphson) is a root-finding algorithm that uses the first term of the Taylor series of the functions f_i to linearise the system (1) in the vicinity of a suspected root. It is one of the oldest and best known methods and is a basis of a number of more refined methods.

Suppose that the point $\mathbf{x} \equiv \{x_1, \ldots, x_n\}$ is close to the root. The Newton's algorithm tries to find the step $\Delta\mathbf{x}$ which would move the point towards the root, such that

$$f_i(\mathbf{x} + \Delta\mathbf{x}) = 0 \ , \ i = 1, \ldots, n \, . \tag{2}$$

The first order Taylor expansion of (2) gives a system of linear equations,

$$f_i(\mathbf{x}) + \sum_{k=1}^{n} \frac{\partial f_i}{\partial x_k} \Delta x_k = 0 \ , \ i = 1, \ldots, n \ , \tag{3}$$

or, in the matrix form,

$$J\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}), \tag{4}$$

where $\mathbf{f}(\mathbf{x}) \equiv \{f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})\}$ and J is the matrix of partial derivatives[1],

$$J_{ik} \equiv \frac{\partial f_i}{\partial x_k} \ , \tag{5}$$

called the *Jacobian matrix*.

The solution $\Delta\mathbf{x}$ to the linear system (4) gives the approximate direction and the step-size towards the solution.

The Newton's method converges quadratically if sufficiently close to the solution. Otherwise the full Newton's step $\Delta\mathbf{x}$ might actually diverge from the solution. Therefore in practice a more conservative step $\lambda\Delta\mathbf{x}$ with $\lambda < 1$ is usually taken. The strategy of finding the optimal λ is referred to as *line search*.

It is typically not worth the effort to find λ which minimizes $\|\mathbf{f}(\mathbf{x} + \lambda\Delta\mathbf{x})\|$ exactly, since $\Delta\mathbf{x}$ is only an approximate direction towards the root. Instead an inexact but quick minimization strategy is usually used, like the *backtracking line search* where one first attempts the full step, $\lambda = 1$, and then backtracks, $\lambda \leftarrow \lambda/2$, until either the condition

$$\|\mathbf{f}(\mathbf{x} + \lambda\Delta\mathbf{x})\| < \left(1 - \frac{\lambda}{2}\right) \|\mathbf{f}(\mathbf{x})\| \tag{6}$$

is satisfied, or λ becomes too small.

[1] in practice if derivatives are not available analytically one uses finite differences

$$\frac{\partial f_i}{\partial x_k} \approx \frac{f_i(x_1, \ldots, x_{k-1}, x_k + \delta x, x_{k+1}, \ldots, x_n) - f_i(x_1, \ldots, x_k, \ldots, x_n)}{\delta x}$$

with $\delta x \ll s$ where s is the typical scale of the problem at hand.

Broyden's quasi-Newton method

The Newton's method requires calculation of the Jacobian at every iteration. This is generally an expensive operation. Quasi-Newton methods avoid calculation of the Jacobian matrix at the new point $\mathbf{x} + \delta\mathbf{x}$, instead trying to use certain approximations, typically rank-1 updates.

Broyden algorithm estimates the Jacobian $J + \delta J$ at the point $\mathbf{x} + \delta\mathbf{x}$ using the finite-difference approximation,

$$(J + \delta J)\delta\mathbf{x} = \delta\mathbf{f} \,, \tag{7}$$

where $\delta\mathbf{f} \equiv \mathbf{f}(\mathbf{x} + \delta\mathbf{x}) - \mathbf{f}(\mathbf{x})$ and J is the Jacobian at the point $\mathbf{x}$.

The matrix equation (7) is under-determined in more than one dimension as it contains only n equations to determine n^2 matrix elements of δJ. Broyden suggested to choose δJ as a rank-1 update, linear in $\delta\mathbf{x}$,

$$\delta J = \mathbf{c}\,\delta\mathbf{x}^T \,, \tag{8}$$

where the unknown vector $\mathbf{c}$ can be found by substituting (8) into (7), which gives

$$\delta J = \frac{\delta\mathbf{f} - J\delta\mathbf{x}}{\|\delta\mathbf{x}\|^2}\delta\mathbf{x}^T \,. \tag{9}$$

Javascript implementation

Optimization

Optimization is a problem of finding the minimum (or the maximum) of a given real (non-linear) function $F(\mathbf{p})$ of an n-dimensional argument $\mathbf{p} \equiv \{x_1, \ldots, x_n\}$.

Downhill simplex method

The *downhill simplex method* (also called Nelder-Mead method or amoeba method) is a commonnly used nonlinear optimization algorithm. The minimum of a function in an n-dimensional space is found by transforming a simplex (a polytope of n+1 vertexes) according to the function values at the vertexes, moving it downhill until it converges towards the minimum.

To introduce the algorithm we need the following definitions:

- Simplex: a figure (polytope) represented by n+1 points, called vertexes, $\{\mathbf{p}_1, \ldots, \mathbf{p}_{n+1}\}$ (where each point $\mathbf{p}_k$ is an n-dimensional vector).
- Highest point: the vertex, $\mathbf{p}_{\rm hi}$, with the largest value of the function: $f(\mathbf{p}_{\rm hi}) = \max_{(k)} f(\mathbf{p}_k)$.
- Lowest point: the vertex, $\mathbf{p}_{\rm lo}$, with the smallest value of the function: $f(\mathbf{p}_{\rm lo}) = \min_{(k)} f(\mathbf{p}_k)$.
- Centroid: the center of gravity of all points, except for the highest: $\mathbf{p}_{\rm ce} = \frac{1}{n}\sum_{(k\neq{\rm hi})}\mathbf{p}_k$

The simplex is moved downhill by a combination of the following elementary operations:

1. Reflection: the highest point is reflected against the centroid, $\mathbf{p}_{\rm hi} \to \mathbf{p}_{\rm re} = \mathbf{p}_{\rm ce} + (\mathbf{p}_{\rm ce} - \mathbf{p}_{\rm hi})$.
2. Expansion: the highest point reflects and then doubles its distance from the centroid, $\mathbf{p}_{\rm hi} \to \mathbf{p}_{\rm ex} = \mathbf{p}_{\rm ce} + 2(\mathbf{p}_{\rm ce} - \mathbf{p}_{\rm hi})$.
3. Contraction: the highest point halves its distance from the centroid, $\mathbf{p}_{\rm hi} \to \mathbf{p}_{\rm co} = \mathbf{p}_{\rm ce} + \frac{1}{2}(\mathbf{p}_{\rm hi} - \mathbf{p}_{\rm ce})$.
4. Reduction: all points, except for the lowest, move towards the lowest points halving the distance. $\mathbf{p}_{k\neq{\rm lo}} \to \frac{1}{2}(\mathbf{p}_k + \mathbf{p}_{\rm lo})$.

Finally, here is a possible algorithm for the downhill simplex method:

Javascript implementation