1 Nonlinear equations

1.1 Introduction

Non-linear equations (or *root-finding*) is a problem of finding a set of n variables $\mathbf{x} \doteq \{x_1, \ldots, x_n\}$ which satisfy a system of n non-linear equations

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

In vector notation the system is written as

$$\mathbf{f}(\mathbf{x}) = 0 \, , \tag{2}$$

where $\mathbf{f}(\mathbf{x}) \doteq \{f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n)\}$.

In one-dimension, $n = 1$, it is generally possible to plot the function in the region of interest and see whether the graph crosses the x-axis. One can then be sure the root exists and even figure out its approximate position to start one's root-finding algorithm from. In multi-dimensions one generally does not know if the root exists at all, until it is found.

The root-finding algorithms generally proceed by iteration, starting from some approximate solution and making consecutive steps—hopefully in the direction of the suspected root—until some convergence criterion is satisfied. The procedure is generally not even guaranteed to converge unless starting from a point close enough to the sought root.

We shall only consider the multi-dimensional case here since i) the multi-dimensional root-finding is more difficult, and ii) the multi-dimensional routines can also be used in the one-dimensional case.

1.2 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 $\mathbf{x} = \{x_1, \ldots, x_n\}$ 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

$$f_i(\mathbf{x} + \Delta\mathbf{x}) = 0 \Big|_{i=1,\ldots,n} . \tag{3}$$

The first order Taylor expansion of (3) gives a system of linear equations,

$$f_i(\mathbf{x}) + \sum_{k=1}^{n} \frac{\partial f_i}{\partial x_k} \Delta x_k = 0 \Big|_{i=1,\ldots,n} , \tag{4}$$

or, in the matrix form,

$$\mathrm{J}\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}), \tag{5}$$

where J is the matrix of partial derivatives,

$$J_{ik} \doteq \frac{\partial f_i}{\partial x_k} \,, \tag{6}$$

called the *Jacobian matrix*. In practice, if derivatives are not available analytically, one uses finite differences,

$$\frac{\partial f_i}{\partial x_k} \approx \frac{f_i(x_1, \ldots, x_k + \delta x_k, \ldots, x_n) - f_i(x_1, \ldots, x_k, \ldots, x_n)}{\delta x_k} \,, \tag{7}$$

where the step δx_k is usually (unless the user knows better) chosen as $\delta x_k = |x_k|\sqrt{\epsilon}$ where ϵ is machine precision. For double-precision numbers $\sqrt{\epsilon} = 2^{-26}$. As a rule of thumb one should always try to rescale one's problem such that the typical scale of one's variables is around unity.

The solution $\Delta\mathbf{x}$ to the linear system (5)—called the Newton's step—gives the approximate direction and the approximate step-size towards the solution.

The Newton's method converges quadratically — that is, the error is squared at each step — if $\mathbf{x}$ is 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, called the *backtracking line search*, where one first attempts the full step, $\lambda = 1$, and then backtracks, $\lambda \leftarrow \lambda/2$, until the condition

$$\|\mathbf{f}(\mathbf{x} + \lambda\Delta\mathbf{x})\| < \left(1 - \frac{\lambda}{2}\right) \|\mathbf{f}(\mathbf{x})\| \tag{8}$$

is satisfied. If the condition is not satisfied for sufficiently small $\lambda_{\min}$ the step is taken with $\lambda_{\min}$ simply to step away from the difficult place and try again.

Here is a typical algrorithm for the Newton's method with backtracking line search and condition (8),

```
repeat
   calculate the Jacobian matrix J
   solve J Δx = −f(x) for Δx
   λ ← 1
   while ‖f(x + λΔx)‖ > (1 − λ/2) ‖f(x)‖ and λ ≥ 1/128 do λ ← λ/2
   x ← x + λΔx
until converged (e.g. ‖f(x)‖ < tolerance)
```

A somewhat more refined backtracking linesearch is based on an approximate minimization of the function

$$\phi(\lambda) \doteq \frac{1}{2}\|\mathbf{f}(\mathbf{x} + \lambda\Delta\mathbf{x})\|^2 \tag{9}$$

using interpolation. The values $\phi(0) = \frac{1}{2}\|\mathbf{f}(\mathbf{x})\|^2$ and $\phi'(0) = -\|\mathbf{f}(\mathbf{x})\|^2$ are already known[1]. If the previous step with certain λ_{trial} was rejected, we also have $\phi(\lambda_{\text{trial}})$. These three quantities allow to build a quadratic approximation,

$$\phi(\lambda) \approx \phi(0) + \phi'(0)\lambda + c\lambda^2 \,, \tag{10}$$

where

$$c = \frac{\phi(\lambda_{\text{trial}}) - \phi(0) - \phi'(0)\lambda_{\text{trial}}}{\lambda_{\text{trial}}^2} \,. \tag{11}$$

The minimum of this approximation (determined by the condition $\phi'(\lambda) = 0$),

$$\lambda_{\text{next}} = -\frac{\phi'(0)}{2c} \,, \tag{12}$$

becomes the next trial step-size.

The procedure is repeated recursively until either condition (8) is satisfied or the step becomes too small (in which case it is taken unconditionally in order to simply get away from the difficult place).

1.3 Quasi-Newton methods

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

1.3.1 Broyden's rank-1 updates

Broyden's algorithms [**?**] estimate the Jacobian $\mathrm{J} + \Delta\mathrm{J}$ at the point $\mathbf{x} + \Delta\mathbf{x}$ using the finite-difference approximation,

$$(\mathrm{J} + \Delta\mathrm{J})\Delta\mathbf{x} = \Delta\mathbf{f} \,, \tag{13}$$

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

Equivalently, one can apply the same approximation for the update $\mathrm{B} + \Delta\mathrm{B}$ of the inverse Jacobian matrix,

$$\Delta\mathbf{x} = (\mathrm{B} + \Delta\mathrm{B})\Delta\mathbf{f} \,, \tag{14}$$

The matrix equation (14) is under-determined in more than one dimension as it contains only n equations to determine n^2 matrix elements of $\Delta\mathrm{B}$. Broyden suggested to choose $\Delta\mathrm{B}$ (or $\Delta\mathrm{J}$) as a rank-1 update that is linear $\Delta\mathbf{x}$. A rank-1

[1] $\frac{\partial\phi}{\partial\lambda}|_{\lambda=0} = \sum_i f_i(\mathbf{x}+\lambda\Delta\mathbf{x})\frac{\partial f_i(\mathbf{x}+\lambda\Delta\mathbf{x})}{\partial\lambda}|_{\lambda=0} = \sum_i f_i \sum_j \frac{\partial f_i(\mathbf{x}+\lambda\Delta\mathbf{x})}{\partial x_j}\Delta x_j|_{\lambda=0} = -\sum_i f_i(\mathbf{x})^2$

update has exactly n free parameters which can be determined form the (inverse secant) equation (14).

For example, one can choose the update in the form

$$\Delta \mathrm{B} = \mathbf{c} \Delta \mathbf{x}^{\mathsf{T}} \,, \tag{15}$$

where $\mathbf{c}$ is an unknown vector. Inserting this ansatz into (14) and solving for $\mathbf{c}$ gives the update

$$\Delta \mathrm{B} = \frac{\Delta \mathbf{x} - \mathrm{B} \Delta \mathbf{f}}{\Delta \mathbf{x}^{\mathsf{T}} \Delta \mathbf{f}} \Delta \mathbf{x}^{\mathsf{T}} \,. \tag{16}$$

Here is a list of several rank-1 updates of the inverse Jacobian matrix,

1. "Good Broyden's method",

$$\Delta \mathrm{B} = \mathbf{c} \Delta \mathbf{x}^{\mathsf{T}} \mathrm{B} \;\Rightarrow\; \Delta \mathrm{B} = \frac{\Delta \mathbf{x} - \mathrm{B} \Delta \mathbf{f}}{\Delta \mathbf{x}^{\mathsf{T}} \mathrm{B} \Delta \mathbf{f}} \Delta \mathbf{x}^{\mathsf{T}} \mathrm{B} \,. \tag{17}$$

2. "Bad Broyden's method",

$$\Delta \mathrm{B} = \mathbf{c} \Delta \mathbf{f}^{\mathsf{T}} \;\Rightarrow\; \Delta \mathrm{B} = \frac{\Delta \mathbf{x} - \mathrm{B} \Delta \mathbf{f}}{\Delta \mathbf{f}^{\mathsf{T}} \Delta \mathbf{f}} \Delta \mathbf{f}^{\mathsf{T}} \,. \tag{18}$$

3. Yet another method,

$$\Delta \mathrm{B} = \mathbf{c} \Delta \mathbf{x}^{\mathsf{T}} \;\Rightarrow\; \Delta \mathrm{B} = \frac{\Delta \mathbf{x} - \mathrm{B} \Delta \mathbf{f}}{\Delta \mathbf{x}^{\mathsf{T}} \Delta \mathbf{f}} \Delta \mathbf{x}^{\mathsf{T}} \,. \tag{19}$$

In practice if one wanders too far from the point where J was first calculated the accuracy of the updates may decrease significantly. In such case one might need to recalculate J anew. For example, two successive steps with $\lambda_{\min}$ might be interpreted as a sign of accuracy loss in J and subsequently trigger its recalculation.

Again, if the denominator in the update formula becomes too small one has to discard the update as it is obviously wrong. If this happens two steps in a row, one has to recalculate one's B-matrix.

Here is a typical quasi-Newton algorithm,

`calculate the inverse Jacobian matrix` $\mathrm{B} = \mathrm{J}^{-1}$
`repeat`
  $\Delta \mathbf{x} = -\mathrm{B}\mathbf{f}(\mathbf{x})$
  $\lambda = 1$
  `while` $\|\mathbf{f}(\mathbf{x} + \lambda \Delta \mathbf{x})\| > \left(1 - \frac{\lambda}{2}\right) \|\mathbf{f}(\mathbf{x})\|$ `and` $\lambda \geq \frac{1}{128}$ `do` $\lambda = \lambda/2$
  $\mathbf{x} = \mathbf{x} + \lambda \Delta \mathbf{x}$
  `if` $\lambda \geq \frac{1}{128}$ `update` $\mathrm{B} = \mathrm{B} + \Delta \mathrm{B}$ `else recalculate` $\mathrm{B} = \mathrm{J}^{-1}$
`until converged (e.g.` $\|\mathbf{f}(\mathbf{x})\| <$ tolerance`)`

The matrix B is updated if the linesearch succeeds, that is, the step-parameter λ is not too small; in case the step-parameter becomes small the step is accepted unconditionally and the B-matrix is recalculated.