ACM40290: Numerical Algorithms

Nonlinear Equations

Dr Barry Wardell
School of Mathematics and Statistics
University College Dublin

Nonlinear Equations

Problem: find the roots, **x**, of a vector-valued function

$$\boxed{\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n \quad \text{so that} \quad \mathbf{f}(\mathbf{x}) = 0}$$

For example, if n=2 we might consider solving the problem

$$\boxed{\mathbf{f} : \begin{cases} x_1 - x_2^2 + 1 = 0 \\ 3x_1 x_2 + x_2^3 = 0 \end{cases}}$$

Nonlinear Equations

We are very familiar with scalar ($n=1$) nonlinear equations

Simplest case is a quadratic equation

$$\boxed{ax^2 + bx + c = 0}$$

We can write down a closed-form solution†, the quadratic formula

$$\boxed{x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

†A closed-form expression involves only a finite number of "well-known" functions, e.g. +, -, ×, ÷, trigonometric functions, logarithms, etc.

Nonlinear Equations

Our n=2 example has similarly trivial solutions

$$\boxed{\begin{aligned} x_1 - x_2^2 + 1 &= 0 \\ 3x_1x_2 + x_2^3 &= 0 \end{aligned}} \Rightarrow \boxed{x_1 = x_2^2 - 1} \Rightarrow \boxed{(4x_2^2 - 3)x_2 = 0}$$

$$\{x_1, x_2\} = \begin{array}{l} \{-1, 0\} \\ \{-\frac{1}{4}, \sqrt{\frac{3}{4}}\} \\ \{-\frac{1}{4}, -\sqrt{\frac{3}{4}}\} \end{array}$$

Nonlinear Equations

What about the more general case?

- In fact, there are closed-form solutions for arbitrary cubic and quartic polynomials (Ferrari & Cardano, ~1540).
- Important mathematical result (Galois, Abel) is that there is no closed-form solution for fifth or higher order polynomial equations.
- Hence, even for the simplest type of nonlinear equation (polynomials on $\mathbb{R}$), the only hope is to employ an iterative algorithm.
- An iterative method should converge in the limit where the number of iterations goes to infinity, and ideally yields an accurate approximation after a small number of iterations.

Nonlinear Equations

Problem: find the roots, **x**, of a vector-valued function

$$\boxed{\mathbf{f}:\mathbb{R}^n \to \mathbb{R}^n \quad \text{so that} \quad \mathbf{f}(\mathbf{x}) = 0}$$

For many real problems *n* may be large, *n*~10,000 is not uncommon.

Use an iterative algorithm to generate a sequence of vectors that converge to the solution.

Minimisation and Optimisation

Closely related to both of these problems is the question of minimising a function

$$g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$$

Such a problem can take one of two forms:

1. Unconstrained optimisation: minimise $g(x)$.
2. Constrained optimisation: minimise $g(x)$ with an additional condition $f(x) = 0$ or possibly $f(x) \geq 0$.

This is a distinct problem, which we will come back to later.

Converting Equations to Fixed Point Form

We wish to solve $f(x) = 0$ iteratively. We need to transform the equation $f(x) = 0$ to the fixed-point form

$$\boxed{x = T(x)}$$

so that we can apply successive approximations which converge to a fixed point of T. Assume

$$\boxed{x_{k+1} = T(x_k)}$$

generates $\{x_0, x_1, x_2, x_3, \ldots, x_k, \ldots\} \to x$.

Converting Equations to Fixed Point Form

Many possible transformations

- $T(x) = x - f(x)$ (Rarely works because T is rarely a contraction mapping)
- $T(x) = x - G(x)\,f(x),\ \ 0 < |G(x)| < \infty$
- etc.

We will look at a set of algorithms that have been well tested and analysed.

Iteratively Solving Nonlinear Equations

We are going to consider iterations of the form:

$$x_{k+1} = T(x_k), \quad k = 0, 1, 2, \ldots \qquad (*)$$

for solving nonlinear equations.

For example, Heron's method for solving $x^2 - a = 0$ (i.e. for computing square roots) is:

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{a}{x_k}\right)$$

This uses $T_{\text{Heron}}(x) = \dfrac{1}{2}\left(x + \dfrac{a}{x}\right)$

Fixed-point Iteration

Suppose α is such that $T(\alpha) = \alpha$, then we call α a **fixed point** of T.

For example, we see that $\sqrt{a}$ is a fixed point of T_{Heron} since

$$T_{\text{Heron}}(\sqrt{a}) = \frac{1}{2}\left(\sqrt{a} + \frac{a}{\sqrt{a}}\right) = \sqrt{a}$$

An iteration of the form (*) "terminates" once a fixed point is reached, since if $T(x_k) = x_k$ then we get $x_{k+1} = x_k$.

Also, if $x_{k+1} = T(x_k)$ converges as $k \to \infty$, it must converge to a fixed point: Let $\alpha \equiv \lim\limits_{k\to\infty} x_k$, then

$$\alpha = \lim_{k\to\infty} x_k = \lim_{k\to\infty} T(x_k) = T\left(\lim_{k\to\infty} x_k\right) = T(\alpha)$$

Fixed-point Iteration

But in order to use an iteration to compute fixed points in practice, we must be able to guarantee that it will converge…

You may recall from other modules that a function T satisfies a *Lipschitz condition* in an interval $[a, b]$ if $\exists L \in \mathbb{R}_{>0}$ such that

$$|T(x) - T(y)| \leq L|x - y|, \quad \forall x, y \in [a, b]$$

If $L < 1$, then T is called a **contraction**.

Fixed-point Iteration

Theorem: Suppose that $T(\alpha) = \alpha$ and that T is a contraction on $[\alpha - A, \alpha + A]$. Suppose also that $|x_0 - \alpha| \leq A$. Then the fixed point iteration converges to α.

Proof:

$$|x_k - \alpha| = |T(x_{k-1}) - T(\alpha)| \leq L|x_{k-1} - \alpha|,$$

which implies

$$|x_k - \alpha| \leq L^k |x_0 - \alpha|$$

and, since $L < 1$, $|x_k - \alpha| \to 0$ as $k \to \infty$.

(Note that $|x_0 - \alpha| \leq A$ implies that all iterates are in $[\alpha - A, \alpha + A]$.

Fixed-point Iteration

Recall that if $T \in C^1[a, b]$, we can obtain a Lipschitz constant based on T':

$$L = \max_{\theta \in (a,b)} |T'(\theta)|$$

We now use this result to show the if $|T'(\alpha)| < 1$, then there is a neighbourhood of α on which T is a contraction.

This tells us that we can verify convergence of a fixed point iteration scheme by checking the gradient of T.

Fixed-point Iteration

By continuity of T' (and hence continuity of $|T'|$), for any $\epsilon > 0$, $\exists\, \delta > 0$ such that for $x \in (\alpha - \delta, \alpha + \delta)$:

$$\big|\, |T'(x)| - |T'(\alpha)| \,\big| \leq \epsilon$$

This implies

$$\max_{x \in (\alpha-\delta, \alpha+\delta)} |T'(x)| \leq |T'(\alpha)| + \epsilon$$

Suppose $|T'(\alpha)| < 1$ and set $\epsilon = \frac{1}{2}\big(1 - |T'(\alpha)|\big)$, then there is a neighbourhood on which T is Lipschitz with $L = \frac{1}{2}\big(1 - |T'(\alpha)|\big)$

Then $L < 1$ and hence ***T* is a contraction in a neighbourhood of α**.

Fixed-point Iteration

Furthermore, as $k \to \infty$,

$$\frac{|x_{k+1} - \alpha|}{|x_k - \alpha|} = \frac{|T(x_k) - T(\alpha)|}{|x_k - \alpha|} \to |T'(\alpha)|$$

Hence, asymptotically, error **decreases by a factor of $|T'(\alpha)|$ after each iteration**.

Order of Convergence

We assume that any iterative algorithm for zero-finding generates a sequence $\{x_0, x_1, \ldots, x_k, \ldots\} \to x$, the solution to $f(x) = 0$, and that the error at stage k is defined as

$$e_k = |x_k - x|, \text{ error at stage } k.$$

Therefore, we may view an iterative algorithm as generating a sequence of errors

$$\{e_0, e_1, \ldots, e_k, \ldots\} \to 0.$$

In theory this sequence is infinite but in practice finite precision forces us to stop at some stage k when $e_k \leq \epsilon$, for some chosen ϵ.

Order of Convergence

Definition (Order of Convergence)
Let $\{x_0, x_1, \ldots, x_k, \ldots\} \to$ x, and $e_k = |x_k - x|$. If there exists a number p and a constant $C \neq 0$ such that

$$\lim_{k\to\infty} \frac{e_k}{e_{k-1}^p} = C$$

then p is called the *Order of Convergence* of $\{x_k\}$.
We assume that $e_k < 1$, for all k. A more usable definition is

$$e_k = Ce_{k-1}^p$$

where C possibly depends on k, but can be bounded above by a constant.
Sequences with $p = 1$ are said to have *Linear Convergence*, while sequences with $p > 1$ have *Super-Linear Convergence*.

Convergence of Successive Approximation

Consider the sequence $\{x_k = T(x_{k\text{-}1})\}$ and assume it converges to a fixed point $x = T(x)$. Assume further that at the fixed point the derivatives $T'(x)$, $T''(x)$, …, $T^{(n)}(x)$ exist. Expanding $T(x_k)$ in a Taylor series about the fixed point x, we get

$$T(x_k) = T(x) + \frac{1}{1!}T'(x)(x_k - x) + \frac{1}{2!}T''(x)(x_k - x)^2 + \cdots + + \frac{1}{n!}T^{(n)}(x)(x_k - x)^n + R_{n+1}$$

or

$$T(x_k) - T(x) = \frac{1}{1!}T'(x)(x_k - x) + \frac{1}{2!}T''(x)(x_k - x)^2 + \cdots + + \frac{1}{n!}T^{(n)}(x)(x_k - x)^n + R_{n+1}$$

Now $|T(x_k) - T(x)| = |x_{k+1} - x| = e_{k+1}$. Hence we get, using the triangle inequality,

$$\boxed{e_{k+1} \leq |T'(x)|e_k + \frac{1}{2}|T''(x)|e_k^2 + \cdots + + \frac{1}{n!}|T^{(n)}(x)|e_k^n + |R_{n+1}|}$$

Convergence of Successive Approximation

1. If $T'(x) \neq 0$ then $e_{k+1} \approx T'(x)\, e_k$, 1st order convergence
2. If $T'(x) = 0$, $T''(x) \neq 0$ then $e_{k+1} \approx T''(x)\, e_k^2$, 2nd order convergence
3. If $T'(x) = T''(x) = \ldots = T^{(n\text{-}1)}(x) = 0$, $T^{(n)}(x) \neq 0$ then $e_{k+1} \approx T^{(n)}(x)\, e_k^n$, n-th order convergence

Example: Square Root $\sqrt{a}$

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{a}{x_k}\right) \equiv T(x_k)$$

The mapping $T(x) = (x + a/x)/2$ has a fixed point $x = \sqrt{a}$. The derivative is $T'(x) = (1-a/x^2)/2$. This gives $T'(\sqrt{a}) = (1-a/a)/2 = 0$. Hence this algorithm has **second order convergence**, at least.

k	0	1	2	3	4	5	6
x_k	1.0	2.5	2.05	2.0006098	2.0000001	2.0000000	2.0000000
e_k	1	0.5	0.05	0.000609	9.2×10^{-8}	2.15×10^{-15}	1.16×10^{-30}

The effect of second order convergence is to double the number of correct digits at each iteration or $e_k \approx e_{k-1}^2 \approx e_0^{2^k}$. If $e_0 = 2^{-1}$ then $e_k \approx e_0^{2^k} = 2^{-2^k}$. For $k = 6$ we get $e_k = 2^{-64} \approx 10^{-20}$. This means that $k = 6$ iterations are sufficient to give full IEEE double precision.

Algorithms for Solving Nonlinear Equations

- Iterative algorithms for non-linear equations fall into two broad categories:
  1. Locally convergent and fast.
  2. Globally convergent and slow.
- A good algorithm should:
  1. Be easy to use, preferably using only information on f, not on its derivative.
  2. Be reliable, i.e., it should find a root close to an initial guess and not go off to become chaotic.
- There is no ideal method. MATLAB uses a combination of methods to find the root. We will study a few of these.

Algorithms for Solving Nonlinear Equations

- Bisection algorithm
- Newton's (Newton-Raphson) method
- Secant algorithm
- Multipoint secant algorithms (e.g. Muller's three-point algorithm, inverse quadratic interpolation)