Math 6610: Analysis of Numerical Methods, I
Numerical solutions of nonlinear equaitons

Today: OH moved
to 3:30–4:30
(WEB 4666)

Department of Mathematics, University of Utah

Fall 2025

Resources: Atkinson 1989, Sections 2.1, 2.2, 2.11
Salgado and Wise 2022, Sections 15.1-15.3

Given $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ a general nonlinear function, consider solving for $\boldsymbol{x}$:

E.g. $\underline{f}(\underline{x}) = \underline{\underline{A}}\,\underline{x} - \underline{b}$

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}.$$

This problem is in general both theoretically and computationally difficult.

- Existence and uniqueness can be difficult to establish
- Iterative algorithms are the typical strategy
- Algorithm success varies wildly depending on the initial iterate, and properties of $\boldsymbol{f}$

Does the algorithm converge? How many iterations?

Does a solution exist? Is there only one?

If $\underline{f}$ is linear/affine: we have bulletproof tools.

Given $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ a general nonlinear function, consider solving for $\boldsymbol{x}$:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}.$$

This problem is in general both theoretically and computationally difficult.

- Existence and uniqueness can be difficult to establish
- Iterative algorithms are the typical strategy
- Algorithm success varies wildly depending on the initial iterate, and properties of $\boldsymbol{f}$

Even with $m = n = 1$ this is a relatively difficult problem.
(E.g., how many solutions should we look for? If $m = n$, is there a single solution?)

There are some standard algorithms for addressing this problem.

We'll only look at a few, but there are numerous methods.

m vs. n doesn't reveal solvability:

$f(\underline{x}) = e^{x_1} + e^{x_2}$

(0 solutions)

$\underline{f}(\underline{x}) = \begin{pmatrix} \sin x \\ \sin(2x) \end{pmatrix}$

(∞-many solutions)

In some cases nonlinear equations can be *linearized*, which informally means that solutions to the nonlinear equation

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0},$$

can be expressed with linear objects and operators.

We've already seen one example of this: eigenvalues.

$\underline{\underline{A}}\,\underline{v} = \lambda \underline{v} \qquad \underline{v} \neq \underline{0}$

$\left.\begin{array}{l} \underline{f}(\underline{v}, \lambda) = \underline{\underline{A}}\,\underline{v} - \lambda \underline{v} = \underline{0} \\ f(\lambda) = \det(\underline{\underline{A}} - \lambda \underline{\underline{I}}) = 0 \end{array}\right\}$ nonlinear rootfinding. But: we can compute solutions through linear algebra (QR, LU, ..)

In some cases nonlinear equations can be *linearized*, which informally means that solutions to the nonlinear equation

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

can be expressed with linear objects and operators.

We've already seen one example of this: eigenvalues.

Another well-known example of linearizations is similar, finding roots of a polynomial:

$$f(x) := x^p + \sum_{j=0}^{p-1} a_j x^j = 0.$$

This is a nonlinear equation for any $p > 1$.

Recall: $(a_0, a_1, \ldots a_{p-1}) \longmapsto f^{-1}(0)$ is generically badly conditioned

$$f(x) := x^p + \sum_{j=0}^{p-1} a_j x^j = 0.$$

Suppose x solves $f(x)=0$

Define $\boldsymbol{C} \in \mathbb{C}^{p\times p}$ by

$$\boldsymbol{C} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{p-1} \end{pmatrix}.$$

$$\underline{v} = \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{p-1} \end{pmatrix} \neq \underline{0}$$

This matrix $\boldsymbol{C}$ is a *companion matrix*.

$$\underline{\underline{C}}\,\underline{v} = \begin{pmatrix} x \\ x^2 \\ \vdots \\ x^{p-1} \\ -\sum_{j=0}^{p-1} a_j x^j \end{pmatrix} = \begin{pmatrix} x \\ x^2 \\ \vdots \\ x^p \end{pmatrix} = x \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{p-1} \end{pmatrix} = x\underline{v}$$

Companion matrix linearizations

A computation shows that if $x_0 \in \mathbb{C}$ is a(ny) root of f, then

$$\boldsymbol{v} = \begin{pmatrix} 1 \\ x_0 \\ x_0^2 \\ \vdots \\ x_0^{p-1} \end{pmatrix}$$

$\underline{\underline{C}}\,\underline{v} = x_0 \underline{v}$

is an eigenvector of $\boldsymbol{C}$ with eigenvalue x_0.
In other words, the spectrum of $\boldsymbol{C}$ is exactly the set of points that solve $f(x) = 0$.

We have numerically stable ways to compute $\Lambda(\underline{\underline{C}})$

Companion matrix linearizations

A computation shows that if $x_0 \in \mathbb{C}$ is a(ny) root of f, then

$$\boldsymbol{v} = \begin{pmatrix} 1 \\ x_0 \\ x_0^2 \\ \vdots \\ x_0^{p-1} \end{pmatrix}$$

is an eigenvector of $\boldsymbol{C}$ with eigenvalue x_0.
In other words, the spectrum of $\boldsymbol{C}$ is exactly the set of points that solve $f(x) = 0$.

(n=m=1)

$$f(x) := x^p + \sum_{j=0}^{p-1} a_j x^j = 0 \quad \Leftrightarrow \quad x \in \Lambda(\boldsymbol{C}).$$

While this provides a way to compute roots via eigenvalue problems, often $\boldsymbol{C}$ is ill-conditioned.
In particular, $\boldsymbol{C}$ is not a normal matrix, so the eigenvalue problem is often poorly conditioned.

This linearization strategy is not really generalizable for $n > 1$, i.e., multivariate polynomials.

There are other versions of companion matrices using different polynomial basis functions: colleague matrices

confederate matrices

⋮

Bisection

A "simple" problem with $f : \mathbb{R} \to \mathbb{R}$:

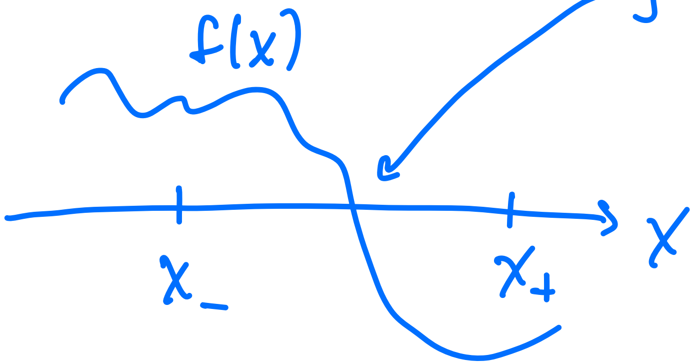
$$f(x) = 0 \qquad (n = 1)$$

Perhaps the simplest numerical method is bisection: assume f is continuous, and that we have two values x_- and x_+ such that

$$x_- < x_+, \qquad\qquad f(x_-)f(x_+) < 0,$$

i.e., $f(x_-)$ and $f(x_+)$ have different signs.
(If one of them is zero, we've already found a root....)

$\exists$ root if f is continuous



Bisection

A "simple" problem with $f : \mathbb{R} \to \mathbb{R}$:

$$f(x) = 0 \qquad (n = 1)$$

Perhaps the simplest numerical method is bisection: assume f is continuous, and that we have two values x_- and x_+ such that

$$x_- < x_+, \qquad\qquad f(x_-)f(x_+) < 0,$$

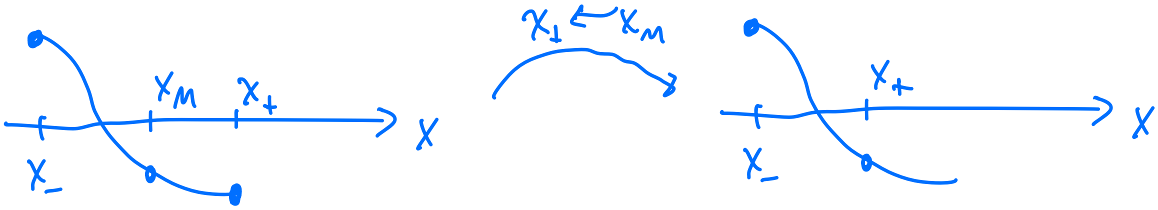
i.e., $f(x_-)$ and $f(x_+)$ have different signs.
(If one of them is zero, we've already found a root....)

In this situation, there must be some solution $x^* \in (x_-, x_+)$ (Intermediate Value Theorem).
The interval (x_-, x_+) is called a *bracketing interval*.

The bisection algorithm zeros in on one solution by progressively creating smaller bracketing intervals:

1. Define $x_M := \frac{1}{2}(x_- + x_-)$, and compute $f(x_M)$.
2. If $f(x_M)f(x_-) < 0$: set $x_+ \leftarrow x_M$ and return to step 1.
3. If $f(x_M)f(x_+) < 0$: set $x_- \leftarrow x_M$ and return to step 1.
4. If $f(x_M) = 0$: then $x^* = x_M$ is the solution.

At any given iteration, any point in the interval, say x_M, is the guess for the root.



x_m
x_+
x_-
x
x_+ ← x_m
x_+
x_-
x

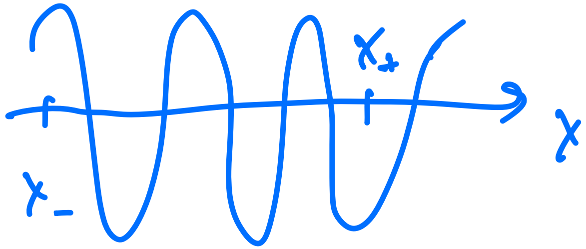
Bisection details

Bisection is quite attractive:

- We require essentially minimal assumptions: just continuity of f
- Exactly and only 1 function evaluation of f per iteration is required
- It's guaranteed to work (provided an initial bracketing interval is identified)

But it has weaknesses:

- There can be several roots inside a bracketing interval – bisection only finds one of them.
- It's relatively slow: convergence is linear, i.e., $|x_{k+1} - x| \leqslant \frac{1}{2}|x_k - x|$.



$$|x_{k+1} - x| \leq \left(\tfrac{1}{2}\right)^k (x_+ - x_-)$$

Stopping criteria

Bisection is a good example to consider an important algorithmic detail: when to stop?

One will generically never identify an x such that $f(x)$ exactly evaluates to 0.

If x_k is the kth iterate (guess for the root), and ϵ_x, ϵ_f are small positive numbers:

- Stop when $|x_{k+1} - x_k| < \epsilon_x$?

E.g.: $f(x) = 10^{10} x$

$|x_1 - x_2| \leq 10^{-6}$

$|f(x_1) - f(x_2)| \leq 10^{4}$

Stopping criteria

Bisection is a good example to consider an important algorithmic detail: when to stop?

One will generically never identify an x such that $f(x)$ exactly evaluates to 0.

If x_k is the kth iterate (guess for the root), and ϵ_x, ϵ_f are small positive numbers:

- Stop when $|x_{k+1} - x_k| < \epsilon_x$?
- Stop when $|x_{k+1} - x_k|/|x_{k+1}| < \epsilon_x$? (Assuming $x_{k+1} \neq 0$....)

Bisection is a good example to consider an important algorithmic detail: when to stop?

One will generically never identify an x such that $f(x)$ exactly evaluates to 0.

If x_k is the kth iterate (guess for the root), and ϵ_x, ϵ_f are small positive numbers:

- Stop when $|x_{k+1} - x_k| < \epsilon_x$?
- Stop when $|x_{k+1} - x_k|/|x_{k+1}| < \epsilon_x$? (Assuming $x_{k+1} \neq 0$....)
- Stop when $|f(x_{k+1}) - f(x_k)| < \epsilon_f$?

Or $|f(x_{k+1})| < \varepsilon_f$

$f(x) = 10^{-10} x$

$x_1 = 1, \quad x_2 = -1$

$|f(x_1)| \leq 10^{-10}$

$|f(x_1) - f(x_2)| \leq 2 \cdot 10^{-10}$

Typically: use some combo of these criteria.

Fixed point iteration, I

A second, more general approach is fixed-point iteration.
Suppose $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$, and we wish to numerically solve,

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}.$$

Fixed point iteration is a computationally simple strategy that rewrites the equation above as

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}),$$

where, for example, $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$.

Fixed point iteration, I

A second, more general approach is fixed-point iteration.
Suppose $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$, and we wish to numerically solve,

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}.$$

Fixed point iteration is a computationally simple strategy that rewrites the equation above as

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}),$$

where, for example, $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$.

Under certain assumptions, the Banach fixed point theorem

- guarantees a unique solution to $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x})$ in a certain neighborhood,
- that the solution is the limit of the sequence $\{\boldsymbol{x}_n\}$ defined by $\boldsymbol{x}_n := \boldsymbol{g}(\boldsymbol{x}_{n-1})$.

Fixed point iteration, I

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}),$$

In order to leverage the Banach fixed point theorem results, $\boldsymbol{g}$ must be a contraction:

- There is some region $D \subseteq \mathbb{R}^n$ such that $\boldsymbol{g} : D \to D$.
- There is some $\lambda \in [0, 1)$ such that $\boldsymbol{g}$ satisfies $\|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{y})\| \leqslant \lambda \|\boldsymbol{x} - \boldsymbol{y}\|$ for every $\boldsymbol{x}, \boldsymbol{y} \in D$.

Fixed point iteration, I

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}),$$

In order to leverage the Banach fixed point theorem results, $\boldsymbol{g}$ must be a contraction:

- There is some region $D \subseteq \mathbb{R}^n$ such that $\boldsymbol{g} : D \to D$.
- There is some $\lambda \in [0, 1)$ such that $\boldsymbol{g}$ satisfies $\|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{y})\| \leqslant \lambda \|\boldsymbol{x} - \boldsymbol{y}\|$ for every $\boldsymbol{x}, \boldsymbol{y} \in D$.

Note that the contraction property is satisfied if, for example,

$$\sup_{\boldsymbol{x} \in D} \left\| \frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\boldsymbol{x}} \right\| < 1,$$

where $\frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\boldsymbol{x}}$ is the Jacobian of $\boldsymbol{g}$.

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}),$$

In order to leverage the Banach fixed point theorem results, $\boldsymbol{g}$ must be a contraction:

- There is some region $D \subseteq \mathbb{R}^n$ such that $\boldsymbol{g} : D \to D$.
- There is some $\lambda \in [0, 1)$ such that $\boldsymbol{g}$ satisfies $\|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{y})\| \leqslant \lambda \|\boldsymbol{x} - \boldsymbol{y}\|$ for every $\boldsymbol{x}, \boldsymbol{y} \in D$.

Note that the contraction property is satisfied if, for example,

$$\sup_{\boldsymbol{x} \in D} \left\| \frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\boldsymbol{x}} \right\| < 1,$$

where $\frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\boldsymbol{x}}$ is the Jacobian of $\boldsymbol{g}$.

Several methods for solving nonlinear equations are variants of fixed point iteration which, given $\boldsymbol{f}$, make special choices for $\boldsymbol{g}$ to ensure the contraction property.

Like bisection, fixed point iteration exhibits linear convergence.
Unlike bisection, fixed point iteration is applicable to n-vector functions of n variables.

Newton's Method ($n = 1$)

A more advanced method is Newton's Method. In the simplest setting, $f : \mathbb{R} \to \mathbb{R}$, we have,

$$f(x) = 0,$$

We cast the problem as the following fixed point iteration:

$$x = g(x) := x - \frac{f(x)}{f'(x)}$$

Note that any solution to $x = g(x)$ also satisfies $f(x) = 0$. (Provided $f'(x) \neq 0$ at the root.)

Newton's Method ($n = 1$)

A more advanced method is Newton's Method. In the simplest setting, $f : \mathbb{R} \to \mathbb{R}$, we have,

$$f(x) = 0,$$

We cast the problem as the following fixed point iteration:

$$x = g(x) := x - \frac{f(x)}{f'(x)}$$

Note that any solution to $x = g(x)$ also satisfies $f(x) = 0$. (Provided $f'(x) \neq 0$ at the root.)

Newton's method applies fixed point iteration:

$$x_n := g(x_{n-1}) = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})},$$

where x_0 must be chosen.
(You've possibly/probably seen alternative motivations for Newton's method, e.g., iteratively finding roots of tangent lines to f.)

Effectiveness of Newton's Method

Newton's Method, under certain assumptions, attains *quadratic* convergence, i.e.,

$$|x - x_n| \leqslant C \, |x - x_{n-1}|^2 ,$$

where x is a root of $f(x)$, and C is an (f, x)-dependent constant.

Effectiveness of Newton's Method

Newton's Method, under certain assumptions, attains *quadratic* convergence, i.e.,

$$|x - x_n| \leqslant C\,|x - x_{n-1}|^2\,,$$

where x is a root of $f(x)$, and C is an (f, x)-dependent constant.

Failure of Newton's Method often results from a poor choice of x_0, or from f not satisfying technical conditions that would ensure success of the method.

When Newton's method fails, it typically fails (numerically) spectacularly.

However, if x_0 is "close enough" to x, then Newton's methods often performs extremely well.

Effectiveness of Newton's Method

Newton's Method, under certain assumptions, attains *quadratic* convergence, i.e.,

$$\left|x - x_n\right| \leqslant C \left|x - x_{n-1}\right|^2,$$

where x is a root of $f(x)$, and C is an (f, x)-dependent constant.

Failure of Newton's Method often results from a poor choice of x_0, or from f not satisfying technical conditions that would ensure success of the method.

When Newton's method fails, it typically fails (numerically) spectacularly.

However, if x_0 is "close enough" to x, then Newton's methods often performs extremely well.

Some methods are hybrids, combining slower and less sophisticated methods, like bisection, to first obtain a guess that is "close" to x.

Subsequently, a faster method, like Newton's Method, is used to converge quickly to the solution.

The Householder methods

There are generalizations of this one-dimensional rootfinding procedure – one family of generalizations are the Householder methods.

Let $f : \mathbb{R} \to \mathbb{R}$ be smooth. For $d \in \mathbb{N}$, the order $(d+1)$ Householder method is the iterative scheme given by,

$$x_{k+1} = x_k + d \frac{\left(\frac{1}{f(x)}\right)^{(d-1)}}{\left(\frac{1}{f(x)}\right)^{(d)}},$$

where $h^{(d)}$ denotes the dth derivative (with respect to x) of h.

There are generalizations of this one-dimensional rootfinding procedure – one family of generalizations are the Householder methods.
Let $f : \mathbb{R} \to \mathbb{R}$ be smooth. For $d \in \mathbb{N}$, the order $(d+1)$ Householder method is the iterative scheme given by,

$$x_{k+1} = x_k + d \frac{\left(\frac{1}{f(x)}\right)^{(d-1)}}{\left(\frac{1}{f(x)}\right)^{(d)}},$$

where $h^{(d)}$ denotes the dth derivative (with respect to x) of h.

Under specific assumptions, this method converges to an exact root $f(x_*) = 0$, with order $d+1$:

$$|x_{k+1} - x_*| \leqslant C \left|x_k - x_*\right|^{d+1},$$

assuming certain properties of f and that x_0 is "close enough" to x_*.

The Householder methods

There are generalizations of this one-dimensional rootfinding procedure – one family of generalizations are the Householder methods.
Let $f : \mathbb{R} \to \mathbb{R}$ be smooth. For $d \in \mathbb{N}$, the order $(d+1)$ Householder method is the iterative scheme given by,

$$x_{k+1} = x_k + d \frac{\left(\frac{1}{f(x)}\right)^{(d-1)}}{\left(\frac{1}{f(x)}\right)^{(d)}},$$

where $h^{(d)}$ denotes the dth derivative (with respect to x) of h.

Under specific assumptions, this method converges to an exact root $f(x_*) = 0$, with order $d+1$:

$$\left|x_{k+1} - x_*\right| \leqslant C \left|x_k - x_*\right|^{d+1},$$

assuming certain properties of f and that x_0 is "close enough" to x_*.

For $d = 1$, this is Newton's method. ($d = 2$ is called *Halley's method*.)

The practical assumptions for large d often outweigh the corresponding convergence gains, unfortunately. (And the larger the d, the more spectacularly these methods fail when they do fail.)

Newton's Method $(n > 1)$

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x} \qquad (m = n > 1)$$

A multivariate form of Newton's Method looks similar to the one-dimensional case:

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}) := \boldsymbol{x} - \left(\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}\right)^{-1} \boldsymbol{f}(\boldsymbol{x}),$$

and the iterates are defined as $\boldsymbol{x}_n = \boldsymbol{g}(\boldsymbol{x}_{n-1})$.

Note in particular that this requires inversion of a (potentially large) matrix at every step.

References I

- Atkinson, Kendall (1989). *An Introduction to Numerical Analysis*. New York: Wiley. ISBN: 978-0-471-62489-9.
- Salgado, Abner J. and Steven M. Wise (2022). *Classical Numerical Analysis: A Comprehensive Course*. Cambridge: Cambridge University Press. ISBN: 978-1-108-83770-5. DOI: 10.1017/9781108942607.