

Numerical Analysis

Solving Nonlinear Equations

Aleksandar Donev
Courant Institute, NYU¹
donev@courant.nyu.edu

¹Course MATH-UA.0252/MA-UY_4424, Spring 2021

Spring 2021

Outline

- 1 Basics of Nonlinear Solvers
- 2 One Dimensional Root Finding
- 3 Systems of Non-Linear Equations

Outline

- 1 Basics of Nonlinear Solvers
- 2 One Dimensional Root Finding
- 3 Systems of Non-Linear Equations

Fundamentals

- Simplest problem: **Root finding** in one dimension:

$$f(x) = 0 \text{ with } x \in [a, b]$$

- Or more generally, solving a **square system of nonlinear equations**

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \quad \Rightarrow \quad f_i(x_1, x_2, \dots, x_n) = 0 \text{ for } i = 1, \dots, n.$$

- There can be no closed-form answer, so just as for eigenvalues, we need **iterative methods**.
- Most generally, starting from $m \geq 1$ **initial guesses** x^0, x^1, \dots, x^m , iterate:

$$x^{k+1} = \phi(x^k, x^{k-1}, \dots, x^{k-m}).$$

Order of convergence

- Consider one dimensional root finding and let the actual root be α , $f(\alpha) = 0$.
- A sequence of iterates x^k that converges to α has **order of convergence** $p \geq 1$ if as $k \rightarrow \infty$

$$\frac{|x^{k+1} - \alpha|}{|x^k - \alpha|^p} = \frac{|e^{k+1}|}{|e^k|^p} \rightarrow C = \text{const},$$

where the constant C is a **convergence factor**, $C < 1$ for $p = 1$.

- A method should at least converge **linearly** ($p = 1$), that is, the error should at least be reduced by a constant factor every iteration, for example, the number of accurate digits increases by 1 every iteration.
- A good method for root finding converges **quadratically** ($p = 2$), that is, the number of accurate digits **doubles** every iteration!

Local vs. global convergence

- A **good initial guess** is extremely important in nonlinear solvers!
- Assume we are looking for a **unique root** $a \leq \alpha \leq b$ starting with an initial guess $a \leq x_0 \leq b$.
- A method has **local convergence** if it converges to a given root α for any initial guess that is sufficiently close to α (in the **neighborhood of a root**).
- A method has **global convergence** if it converges to the root for any initial guess.
- General rule: Global convergence requires a **slower** (careful) method **but is safer**.
- It is best to combine a global method to first find a good initial guess close to α and then use a faster local method.

Conditioning of root finding

$$f(\alpha + \delta\alpha) \approx f(\alpha) + f'(\alpha)\delta\alpha = \delta f$$

$$|\delta\alpha| \approx \frac{|\delta f|}{|f'(\alpha)|} \Rightarrow \kappa_{abs} = |f'(\alpha)|^{-1}.$$

- The problem of finding a **simple root** is well-conditioned when $|f'(\alpha)|$ is far from zero.
- Finding **roots with multiplicity** $m > 1$ is **ill-conditioned**:

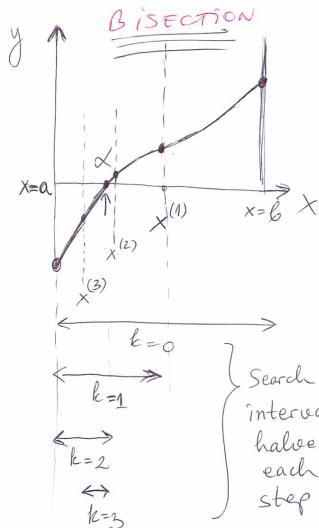
$$|f'(\alpha)| = \dots = |f^{(m-1)}(\alpha)| = 0 \Rightarrow |\delta\alpha| \approx \left[\frac{|\delta f|}{|f^m(\alpha)|} \right]^{1/m}$$

- Note that finding **roots of algebraic equations** (polynomials) is a separate subject of its own that we skip.

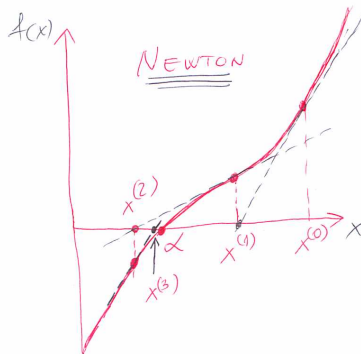
Outline

- 1 Basics of Nonlinear Solvers
- 2 One Dimensional Root Finding**
- 3 Systems of Non-Linear Equations

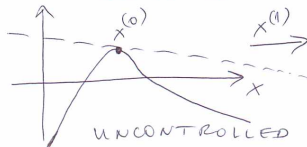
The bisection and Newton algorithms



GLOBAL SLOW CONVERGENCE



LOCAL FAST CONVERGENCE



Bisection

- First step is to **locate a root** by searching for a **sign change**, i.e., finding a^0 and b^0 such that

$$f(a^0)f(b^0) < 0.$$

- The simply **bisect** the interval, for $k = 0, 1, \dots$

$$x^k = \frac{a^k + b^k}{2}$$

and choose the half in which the function changes sign, i.e., either $a^{k+1} = x^k$, $b^{k+1} = b^k$ or $b^{k+1} = x^k$, $a^{k+1} = a^k$ so that $f(a^{k+1})f(b^{k+1}) < 0$.

- Observe that each step we need one **function evaluation**, $f(x^k)$, but only the sign matters.
- The **convergence is essentially linear** because

$$|x^k - \alpha| \leq \frac{b^k}{2^{k+1}} \quad \Rightarrow \quad \frac{|x^{k+1} - \alpha|}{|x^k - \alpha|} \leq 2.$$

Newton's Method

- Bisection is a slow but sure method. It uses no information about the value of the function or its derivatives.
- Better convergence, of order $p = (1 + \sqrt{5})/2 \approx 1.63$ (the golden ratio), can be achieved by using the value of the function at two points, as in the **secant method**.
- Achieving second-order convergence requires also evaluating the **function derivative**.
- **Linearize the function** around the current guess using Taylor series:

$$f(x^{k+1}) \approx f(x^k) + (x^{k+1} - x^k)f'(x^k) = 0$$

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

Convergence of Newton's method

Use Taylor series with remainder and divide by $f'(x^k) \neq 0$:

$$\exists \xi \in [x_n, \alpha] : f(\alpha) = 0 = f(x^k) + (\alpha - x^k)f'(x^k) + \frac{1}{2}(\alpha - x^k)^2 f''(\xi) = 0,$$

$$\left[x^k - \frac{f(x^k)}{f'(x^k)} \right] - \alpha = -\frac{1}{2}(\alpha - x^k)^2 \frac{f''(\xi)}{f'(x^k)}$$

$$x^{k+1} - \alpha = e^{k+1} = -\frac{1}{2} (e^k)^2 \frac{f''(\xi)}{f'(x^k)}$$

which shows **second-order convergence**

$$\frac{|x^{k+1} - \alpha|}{|x^k - \alpha|^2} = \frac{|e^{k+1}|}{|e^k|^2} = \left| \frac{f''(\xi)}{2f'(x^k)} \right| \rightarrow \left| \frac{f''(\alpha)}{2f'(\alpha)} \right|$$

Basin of attraction for Newton's method

- For convergence we want $|e^{k+1}| < |e^k|$ so we want

$$|e^k| \left| \frac{f''(\alpha)}{2f'(\alpha)} \right| < 1 \quad \Rightarrow \quad |e^k| < \left| \frac{2f'(\alpha)}{f''(\alpha)} \right|$$

- Newton's method **converges quadratically** if we start **sufficiently close to a simple root**, more precisely, if

$$|x^0 - \alpha| = |e^0| \lesssim \left| \frac{2f'(\alpha)}{f''(\alpha)} \right|.$$

This is just a **rough estimate**, not a precise bound!

- A robust but fast algorithm for root finding would **safeguard Newton's method** with bisection: Eventually we will accept all Newton steps once close to the root, so we will get **quadratic convergence**, but also be **guaranteed to converge** to a root.

Fixed-Point Iteration

$$f(x) = 0 \quad \Rightarrow \quad x = f(x) + x = \phi(x)$$

- Then we can use **fixed-point iteration**

$$x^{k+1} = \phi(x^k)$$

- whose fixed point (limit), if it converges, is $x \rightarrow \alpha$. Taylor series estimate:

$$x^{k+1} = \alpha + e^{k+1} \approx \phi(\alpha) + \phi'(\alpha) (x^k - \alpha) = \alpha + \phi'(\alpha)e^k \quad \Rightarrow$$

$$e^{k+1} \approx \phi'(\alpha)e^k \quad \Rightarrow \quad \text{we want } |\phi'(\alpha)| < 1.$$

- It can be proven that the fixed-point iteration converges if $\phi(x)$ is a **contraction mapping**:

$$|\phi'(x)| \leq K < 1 \quad \forall x \in [a, b]$$

[If $\phi(x)$ is Lipschitz continuous with Lipschitz constant $L < 1$.]

Stopping Criteria

- A good library function for root finding has to implement careful termination criteria.
- An obvious option is to terminate when the **residual becomes small**

$$|f(x^k)| < \varepsilon,$$

which **works for very well-conditioned problems**, $|f'(\alpha)| \sim 1$.

- Another option is to terminate when the **increment becomes small**

$$|x^{k+1} - x^k| < \varepsilon.$$

- For example, for fixed-point iteration this test would stop at step k :

$$x^{k+1} - x^k = e^{k+1} - e^k \approx [1 - \phi'(\alpha)] e^k \quad \Rightarrow \quad |e^k| \approx \frac{\varepsilon}{|1 - \phi'(\alpha)|},$$

so we see that the increment test **works for rapidly converging iterations**, i.e., when $|1 - \phi'(\alpha)|$ is not small.

In practice

- A robust but fast algorithm for root finding would **combine (safeguard) bisection with Newton's method**: Given a current bisection interval $[a, b]$, if $x^{k+1} \in (a, b)$ then accept Newton step, otherwise just set $x^{k+1} = (a + b)/2$. Take new bisection interval to be either $[a, x^{k+1}]$ or $[x^{k+1}, b]$ the same way as in bisection where we always use $x^{k+1} = (a + b)/2$.
- Newton's method requires first-order derivatives so often other methods are preferred that require **function evaluation only**. Examples include secant method (based on linear interpolation) or **inverse quadratic interpolation** (fit a parabola through three past points $(f(x_i), x_i)$, $i = 1, 2, 3$, and evaluate for zero argument to give a new point).
- Matlab's function `fzero` combines bisection, secant and inverse quadratic interpolation and is "fail-safe". See, for example, "**Brent's method**" on Wikipedia.

Find zeros of $a \sin(x) + b \exp(-x^2/2)$

```

% f=@mfile uses a function in an m-file

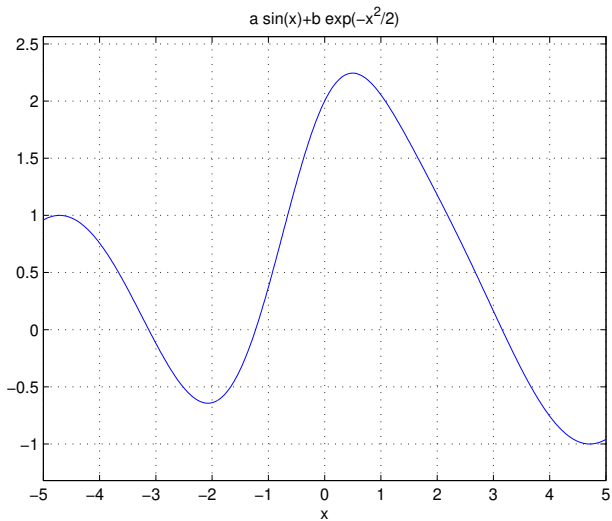
% Parameterized functions are created with:
a = 1; b = 2;
f = @(x) a*sin(x) + b*exp(-x^2/2) ; % Handle

figure(1)
ezplot(f,[-5,5]); grid

x1=fzero(f, [-2,0])
[x2,f2]=fzero(f, 2.0)

x1 = -1.227430849357917
x2 = 3.155366415494801
f2 = -2.116362640691705e-16

```

Figure of $f(x)$ 

Outline

- 1 Basics of Nonlinear Solvers
- 2 One Dimensional Root Finding
- 3 Systems of Non-Linear Equations**

Multi-Variable Taylor Expansion

- It is convenient to focus on one of the equations, i.e., consider a **scalar function** $f(\mathbf{x})$.
- The usual Taylor series is replaced by

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) + \mathbf{g}^T (\Delta\mathbf{x}) + \frac{1}{2} (\Delta\mathbf{x})^T \mathbf{H} (\Delta\mathbf{x})$$

where the **gradient vector** is

$$\mathbf{g} = \nabla_{\mathbf{x}} f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right]^T$$

and the **Hessian matrix** is

$$\mathbf{H} = \nabla_{\mathbf{x}}^2 f = \left\{ \frac{\partial^2 f}{\partial x_i \partial x_j} \right\}_{ij}$$

Vector Functions of Vectors

- We are after solving a **square system of nonlinear equations** for some variables \mathbf{x} :

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \quad \Rightarrow \quad f_i(x_1, x_2, \dots, x_n) = 0 \quad \text{for } i = 1, \dots, n.$$

- The first-order Taylor series is

$$\mathbf{f}(\mathbf{x}^k + \Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^k) + [\mathbf{J}(\mathbf{x}^k)] \Delta\mathbf{x} = \mathbf{0}$$

where the Jacobian \mathbf{J} has the gradients of $f_i(\mathbf{x})$ as rows:

$$[\mathbf{J}(\mathbf{x})]_{ij} = \frac{\partial f_i}{\partial x_j}$$

Newton's Method for Systems of Equations

- It is much harder if not impossible to do globally convergent methods like bisection in higher dimensions!
- A good initial guess is therefore a must when solving systems, and Newton's method can be used to refine the guess.
- The basic idea behind Newton's method is to **linearize** the equation around the current guess:

$$\mathbf{f}(\mathbf{x}^k + \Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^k) + [\mathbf{J}(\mathbf{x}^k)] \Delta\mathbf{x} = \mathbf{0}$$

$$[\mathbf{J}(\mathbf{x}^k)] \Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}^k) \quad \text{but denote } \mathbf{J} \equiv \mathbf{J}(\mathbf{x}^k)$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x} = \mathbf{x}^k - \mathbf{J}^{-1}\mathbf{f}(\mathbf{x}^k).$$

- This method requires computing a whole **matrix of derivatives**, which can be expensive or hard to do (differentiation by hand?)!

Convergence of Newton's method

- Near the root the Jacobian and Hessian don't change much so just approximate $\mathbf{J} \approx \mathbf{J}(\alpha)$ and $\mathbf{H} \approx \mathbf{H}(\alpha)$.
- Next order term in Taylor series indicates error

$$\mathbf{f}(\mathbf{x}^k) = \mathbf{f}(\alpha) + \mathbf{J}\mathbf{e}^k + \frac{1}{2}(\mathbf{e}^k)^T \mathbf{H}\mathbf{e}^k = \mathbf{J}\mathbf{e}^k + \frac{1}{2}(\mathbf{e}^k)^T \mathbf{H}\mathbf{e}^k \quad \Rightarrow$$

$$\mathbf{e}^{k+1} = \mathbf{x}^{k+1} - \alpha = \mathbf{e}^k - \mathbf{J}^{-1}\mathbf{f}(\mathbf{x}^k) = \frac{1}{2}\mathbf{J}^{-1}(\mathbf{e}^k)^T \mathbf{H}\mathbf{e}^k$$

- Newton's method converges **quadratically** if started sufficiently close to a root α :

$$\|\mathbf{e}^{k+1}\| \leq \frac{\|\mathbf{J}^{-1}\| \|\mathbf{H}\|}{2} \|\mathbf{e}^k\|^2$$

- Newton's method converges fast **if the Jacobian $\mathbf{J}(\alpha)$ is well-conditioned.**
- Newton's method requires solving **many linear systems**, which can be expensive for many variables.

Conclusions/Summary

- Root finding is well-conditioned for **simple roots** (unit multiplicity), ill-conditioned otherwise.
- Methods for solving nonlinear equations are always iterative and the **order of convergence** matters: second order is usually good enough.
- A good method uses a higher-order unsafe method such as **Newton method** near the root, but **safeguards** it with something like the **bisection** method.
- Newton's method is second-order but requires derivative/Jacobian evaluation. In **higher dimensions** having a **good initial guess** for Newton's method becomes very important.