Numerical Methods I
Solving Nonlinear Equations

Aleksandar Donev
Courant Institute, NYU[1]
donev@courant.nyu.edu

[1] Course G63.2010.001 / G22.2420-001, Fall 2010

October 14th, 2010

Outline

Final Presentations

- The final project writeup will be due Sunday Dec. 26th by midnight (I have to start grading by 12/27 due to University deadlines).
- You will also need to give a 15 minute presentation in front of me and other students.
- Our last class is officially scheduled for Tuesday 12/14, 5-7pm, and the final exam Thursday 12/23, 5-7pm. Neither of these are good!
- By the end of next week, October 23rd, please let me know the following:
  - Are you willing to present early Thursday December 16th during usual class time?
  - Do you want to present during the official scheduled last class, Thursday 12/23, 5-7pm.
  - If neither of the above, tell me when you **cannot** present Monday Dec. 20th to Thursday Dec. 23rd (finals week).

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**

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

- 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** $x^0, x^1, \ldots, x^m$, iterate:

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

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** $p > 1$ if as $k \to \infty$

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

  where the constant $0 < C < 1$ is the **convergence factor**.
- A method should at least converge **linearly**, 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 coverges **quadratically**, 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

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

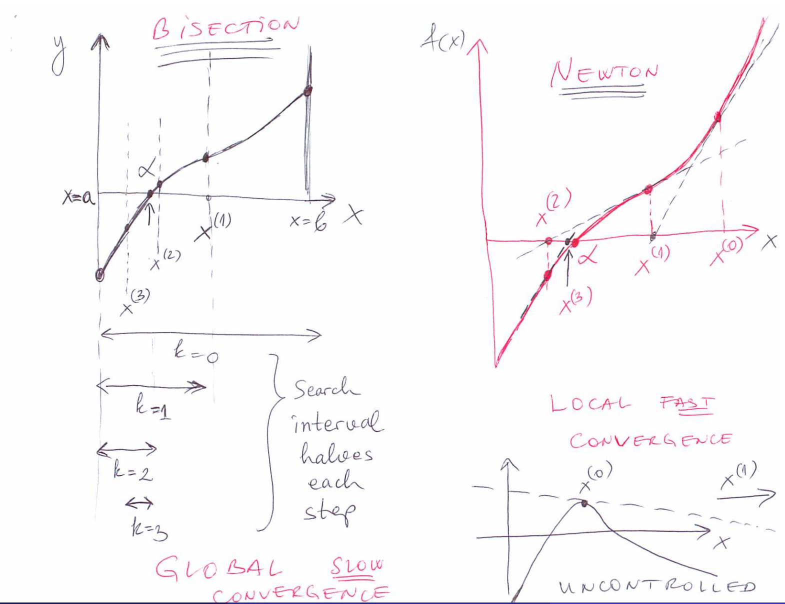
$$|\delta\alpha| \approx \frac{|\delta f|}{|f'(\alpha)|} \quad \Rightarrow \kappa_{abs} = \left|f'(\alpha)\right|^{-1}.$$

- 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**:

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

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

The bisection and Newton algorithms



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,

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

and choose the half in which the function changes sign by looking at the sign of $f(x^k)$.

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

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

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

Taylor series with remainder:

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

After dividing by $f'(x^k) \neq 0$ we get

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

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

which shows **second-order convergence**

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

Proof of Local Convergence

$$\frac{\left|x^{k+1}-\alpha\right|}{\left|x^{k}-\alpha\right|^{2}}=\left|\frac{f''(\xi)}{2f'(x^{k})}\right|\leq M=\sup_{\alpha-|e^{0}|\leq x,y\leq\alpha+|e^{0}|}\left|\frac{f''(x)}{2f'(y)}\right|$$

$$M\left|x^{k+1}-\alpha\right|=E^{k+1}\leq\left(M\left|x^{k}-\alpha\right|\right)^{2}=\left(E^{k}\right)^{2}$$

which will converge if $E^{0}<1$, i.e., if

$$\left|x^{0}-\alpha\right|=\left|e^{0}\right|<M^{-1}$$

Newton's method thus always converges quadratically if we start sufficiently close to a simple root.

Fixed-Point Iteration

- Another way to devise iterative root finding is to rewrite $f(x)$ in an equivalent form

$$x = \phi(x)$$

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

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

  whose fixed point (limit), if it converges, is $x \to \alpha$.

- For example, recall from first lecture solving $x^2 = c$ via the Babylonian method for square roots

$$x_{n+1} = \phi(x_n) = \frac{1}{2}\left(\frac{c}{x} + x\right),$$

  which converges (quadratically) for any non-zero initial guess.

Convergence theory

- It can be proven that the fixed-point iteration $x^{k+1} = \phi(x^k)$ converges if $\phi(x)$ is a **contraction mapping**:

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

$$x^{k+1} - \alpha = \phi(x^k) - \phi(\alpha) = \phi'(\xi)\left(x^k - \alpha\right) \text{ by the mean value theorem}$$

$$\left|x^{k+1} - \alpha\right| < K\left|x^k - \alpha\right|$$

- If $\phi'(\alpha) \neq 0$ near the root we have **linear convergence**

$$\frac{\left|x^{k+1} - \alpha\right|}{\left|x^k - \alpha\right|} \to \phi'(\alpha).$$

- If $\phi'(\alpha) = 0$ we have **second-order convergence** if $\phi''(\alpha) \neq 0$, etc.

Applications of general convergence theory

- Think of Newton's method

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

as a fixed-point iteration method $x^{k+1} = \phi(x^k)$ with iteration function:

$$\phi(x) = x - \frac{f(x)}{f'(x)}.$$

- We can directly show quadratic convergence because (also see homework)

$$\phi'(x) = \frac{f(x)f''(x)}{[f'(x)]^2} \quad \Rightarrow \quad \phi'(\alpha) = 0$$

$$\phi''(\alpha) = \frac{f''(\alpha)}{f'(\alpha)} \neq 0$$

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**

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

which is only good for very well-conditioned problems, $|f'(\alpha)| \sim 1$.
- Another option is to terminate when the **increment becomes small**

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

- For fixed-point iteration

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

so we see that the increment test works for rapidly converging iterations ($\phi'(\alpha) \ll 1$).

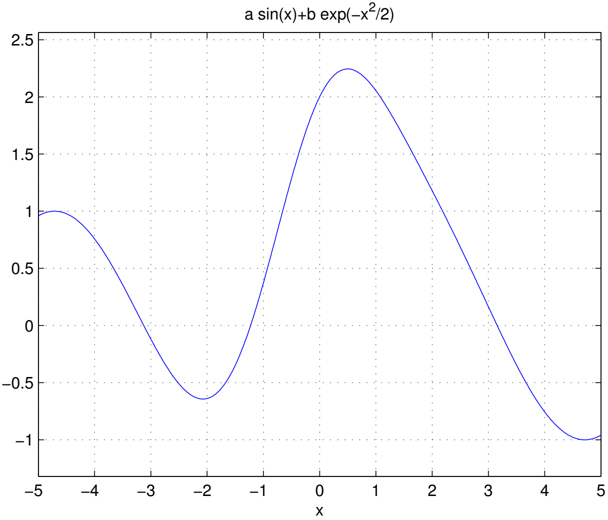
In practice

- A robust but fast algorithm for root finding would **combine bisection with Newton's method**.
- Specifically, a method like Newton's that can easily take huge steps in the wrong direction and lead far from the current point must be **safeguarded** by a method that ensures one does not leave the search interval and that the zero is not missed.
- Once x^k is close to α, the safeguard will not be used and quadratic or faster convergence will be achieved.
- Newton's method requires first-order derivatives so often other methods are preferred that require **function evaluation only**.
- Matlab's function *fzero* combines bisection, secant and inverse quadratic interpolation and is "fail-safe".

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

```
% 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 $f(x)$



Multi-Variable Taylor Expansion

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

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

- 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

$$\mathbf{g} = \boldsymbol{\nabla}_{\mathbf{x}} f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right]^T$$

and the **Hessian matrix** is

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

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 first-order Taylor series is

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

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

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

- So taking a Newton step requires solving a linear system:

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

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

Convergence of Newton's method

- Newton's method converges **quadratically** if started sufficiently close to a root $\mathbf{x}^{\star}$ at which the Jacobian is not singular.

$$\left\|\mathbf{x}^{k+1}-\mathbf{x}^{\star}\right\| = \left\|\mathbf{e}^{k+1}\right\| = \left\|\mathbf{x}^{k}-\mathbf{J}^{-1}\mathbf{f}\left(\mathbf{x}^{k}\right)-\mathbf{x}^{\star}\right\| = \left\|\mathbf{e}^{k}-\mathbf{J}^{-1}\mathbf{f}\left(\mathbf{x}^{k}\right)\right\|$$

but using second-order Taylor series

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

$$\left\|\mathbf{e}^{k+1}\right\| = \left\|\frac{\mathbf{J}^{-1}}{2}\left(\mathbf{e}^{k}\right)^{T}\mathbf{H}\left(\mathbf{e}^{k}\right)\right\| \leq \frac{\left\|\mathbf{J}^{-1}\right\|\left\|\mathbf{H}\right\|}{2}\left\|\mathbf{e}^{k}\right\|^{2}$$

- Fixed point iteration theory generalizes to multiple variables, e.g., replace $f'(\alpha) < 1$ with $\rho\left(\mathbf{J}(\mathbf{x}^{\star})\right) < 1$.

Problems with Newton's method

- Newton's method requires solving **many linear systems**, which can become complicated when there are many variables.
- It also requires computing a whole **matrix of derivatives**, which can be expensive or hard to do (differentiation by hand?)
- Newton's method converges fast if the Jacobian $\mathbf{J}(\mathbf{x}^{\star})$ is well-conditioned, otherwise it can "blow up".
- For large systems one can use so called **quasi-Newton** methods:
  - Approximate the Jacobian with another matrix $\widetilde{\mathbf{J}}$ and solve $\widetilde{\mathbf{J}}\Delta\mathbf{x} = \mathbf{f}(\mathbf{x}^k)$.
  - Damp the step by a step length $\alpha_k \lesssim 1$,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \Delta\mathbf{x}.$$

  - Update $\widetilde{\mathbf{J}}$ by a simple update, e.g., a rank-1 update (recall homework 2).

In practice

- It is much harder to construct general robust solvers in higher dimensions and some **problem-specific knowledge** is required.
- There is no built-in function for solving nonlinear systems in MATLAB, but the **Optimization Toolbox** has *fsolve*.
- In many practical situations there is some continuity of the problem so that **a previous solution can be used as an initial guess**.
- For example, **implicit methods for differential equations** have a time-dependent Jacobian $\mathbf{J}(t)$ and in many cases the solution $\mathbf{x}(t)$ evolves smooty in time.
- For large problems specialized **sparse-matrix solvers** need to be used.
- In many cases derivatives are not provided but there are some techniques for **automatic differentiation**.

Formulation

- Optimization problems are among the most important in engineering and finance, e.g., **minimizing** production cost, **maximizing** profits, etc.

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

  where $\mathbf{x}$ are some variable parameters and $f : \mathbb{R}^n \to \mathbb{R}$ is a scalar **objective function**.

- Observe that one only need to consider minimization as

$$\max_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = -\min_{\mathbf{x}\in\mathbb{R}^n} \left[-f(\mathbf{x})\right]$$

- A **local minimum** $\mathbf{x}^\star$ is optimal in some neighborhood,

$$f(\mathbf{x}^\star) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \quad \text{s.t.} \quad \|\mathbf{x}-\mathbf{x}^\star\| \leq R > 0.$$

  (think of finding the bottom of a valley)

- Finding the **global minimum** is generally not possible for *arbitrary* functions

  (think of finding Mt. Everest without a satelite)

Connection to nonlinear systems

- Assume that the objective function is **differentiable** (i.e., first-order Taylor series converges or gradient exists).
- Then a **necessary condition** for a local minimizer is that $\mathbf{x}^{\star}$ be a **critical point**

$$\mathbf{g}\left(\mathbf{x}^{\star}\right)=\boldsymbol{\nabla}_{\mathbf{x}} f\left(\mathbf{x}^{\star}\right)=\left\{\frac{\partial f}{\partial x_i}\left(\mathbf{x}^{\star}\right)\right\}_i=\mathbf{0}$$

  which is a system of non-linear equations!
- In fact similar methods, such as Newton or quasi-Newton, apply to both problems.
- Vice versa, observe that solving $\mathbf{f}\left(\mathbf{x}\right)=\mathbf{0}$ is equivalent to an optimization problem

$$\min_{\mathbf{x}}\left[\mathbf{f}\left(\mathbf{x}\right)^{T}\mathbf{f}\left(\mathbf{x}\right)\right]$$

  although this is only recommended under special circumstances.

Sufficient Conditions

- Assume now that the objective function is **twice-differentiable** (i.e., Hessian exists).
- A critical point $\mathbf{x}^{\star}$is a local minimum if the **Hessian is positive definite**

$$\mathbf{H}\left(\mathbf{x}^{\star}\right)=\boldsymbol{\nabla}_{\mathbf{x}}^{2} f\left(\mathbf{x}^{\star}\right) \succ \mathbf{0}$$

  which means that the minimum really looks like a valley or a **convex** bowl.
- At any local minimum the Hessian is positive **semi-definite**, $\boldsymbol{\nabla}_{\mathbf{x}}^{2} f\left(\mathbf{x}^{\star}\right) \succeq \mathbf{0}$.
- Methods that require Hessian information converge fast but are expensive (next class).

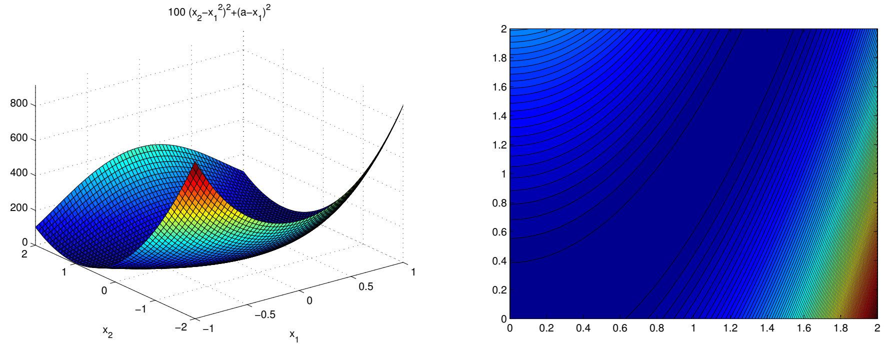
Direct-Search Methods

- A **direct search method** only requires $f(\mathbf{x})$ to be **continuous** but not necessarily differentiable, and requires only **function evaluations**.
- Methods that do a search similar to that in bisection can be devised in higher dimensions also, but they may fail to converge and are usually slow.
- The MATLAB function *fminsearch* uses the Nelder-Mead or **simplex-search** method, which can be thought of as rolling a simplex downhill to find the bottom of a valley. But there are many others and this is an active research area.
- **Curse of dimensionality**: As the number of variables (dimensionality) n becomes larger, direct search becomes hopeless since the number of samples needed grows as 2^n!

Minimum of $100(x_2 - x_1^2)^2 + (a - x_1)^2$ in MATLAB

```
% Rosenbrock or 'banana' function:
a = 1;
banana = @(x) 100*(x(2)-x(1)^2)^2+(a-x(1))^2;

% This function must accept array arguments!
banana_xy = @(x1,x2) 100*(x2-x1.^2).^2+(a-x1).^2;

figure(1); ezsurf(banana_xy, [0,2,0,2])

[x,y] = meshgrid(linspace(0,2,100));
figure(2); contourf(x,y,banana_xy(x,y),100)

% Correct answers are x=[1,1] and f(x)=0
[x,fval] = fminsearch(banana, [-1.2, 1], optimset('TolX',1e-8))
x =     0.999999999187814     0.999999998441919
fval =       1.099088951919573e-18
```

Figure of Rosenbrock $f(\mathbf{x})$



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.
- **Quasi-Newton** methods can aleviate the complexity of solving the Jacobian linear system.