CS 450: Numerical Analysis
Nonlinear Equations

University of Illinois at Urbana-Champaign

---

These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book “Scientific Computing: An Introductory Survey” by Michael T. Heath (slides).
Solving Nonlinear Equations

- Solving (systems of) nonlinear equations corresponds to root finding:
  - $f(x^*) = 0$ (single-variate)
  - $f(x^*) = 0$ (multi-variate, scalar valued)
  - $f(x^*) = 0$ (vector-valued)

- Algorithms for root-finding make it possible to solve systems of nonlinear equations and employ a similar methodology to finding minima in optimization.

- Main algorithmic approach: find successive roots of local linear approximations of $f$:

  **Newton's method**
  
  $f(x^*) = 0$
  
  $f(x_k + s) = f(x_k) + s \cdot f'(x_k) = 0$

  \[ s_k = -f(x_k) / f'(x_k) \]

  $x_{k+1} = x_k + s_k = x_k - f(x_k) / f'(x_k)$

**Activity: Newton's Method for 2-by-2 System of Equations**
\[ \text{Jacobian } \quad J_f(x) = \begin{bmatrix} \frac{df_1}{dx_1} & \ldots & \frac{df_1}{dx_m} \\ \vdots & \ddots & \vdots \\ \frac{df_n}{dx_1} & \ldots & \frac{df_n}{dx_m} \end{bmatrix} \quad f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix} \]

\[ f(x^k + \delta x) \approx f(x^k) + J_f(x^k) \delta x = 0 \]

\[ \begin{align*}
J_f(x^k) \delta x &= -f(x^k) \\
\delta x &= -J_f(x^k)^{-1} f(x^k)
\end{align*} \]
Nonexistence and Nonuniqueness of Solutions

- Solutions do not generally exist and are not generally unique, even in the univariate case:

- Solutions in the multivariate case correspond to intersections of hypersurfaces:

*Demo: Three quadratic functions*
Conditions for Existence of Solution

- **Intermediate value theorem** for univariate problems:
  \[ \text{bracketed } [a, b], \ \text{sign } (f(a)) \neq \text{sign } (f(b)) \]

- A function has a unique **fixed point** \( g(x^*) = x^* \) in a given closed domain if it is **contractive** and contained in that domain,

\[
\begin{align*}
g(x) &= f(x) + x \\
g(x^*) &= x^* \\
\Rightarrow f(x^*) &= 0
\end{align*}
\]

\[ ||g(x) - g(z)|| \leq \gamma ||x - z|| \]

**Lipschitz constant**
Conditioning of Nonlinear Equations

- Generally, we take interest in the absolute rather than relative conditioning of solving $f(x) = 0$:

- The absolute condition number of finding a root $x^*$ of $f$ is $1/|f'(x^*)|$ and for a root $x^*$ of $f$ it is $||J_f^{-1}(x^*)||$:

  \[ \kappa_{abs}(\text{root finding for } f \text{ for root } x^*) = \frac{1}{f'(x^*)} \]
Multiple Roots and Degeneracy

- If $x^*$ is a root of $f$ with multiplicity $m$, its $m - 1$ derivatives are also zero at $x^*$,

$$f(x^*) = f'(x^*) = f''(x^*) = \cdots = f^{(m-1)}(x^*) = 0.$$ 

$$f(x) = (x - x^*)^m h(x)$$ 

$$f'(x) = (x - x^*)^{m-1} h(x) + (x - x^*)^m h'(x)$$ 

$$= (x - x^*)^{m-1} \left( h(x) + (x - x^*) h'(x) \right)$$

- Increased multiplicity affects conditioning and convergence:

$m > 1 \Rightarrow$ ill-posed

Can modify problem/algorithm to handle $m > 1$
Bisection Algorithm

- Assume we know the desired root exists in a bracket \([a, b]\) and \(\text{sign}(f(a)) \neq \text{sign}(f(b))\):

- Bisection subdivides the interval by a factor of two at each step by considering \(f(c_k)\) at \(c_k = (a_k + b_k)/2\):
Let $x_k$ be the $k$th iterate and $e_k = x_k - x^*$ be the error, bisection obtains linear convergence, $\lim_{k \to \infty} \frac{||e_k||}{||e_{k-1}||} \leq C$:

at each step, we gain $O(1)$ digits of accuracy for bisection

$e_k / e_{k-1} \leq 1/2$

$r$th order convergence implies that $||e_k|| / ||e_{k-1}||^r \leq C$

$r = 1$ superlinear
$r = 2$ quadratic
$r = 3$ cubic

number of digits of accuracy improves by factor of $r$

$O(\log(\log(1/\varepsilon)))$ steps
changing $r$, changes complexity by a constant
Convergence of Fixed Point Iteration

- Fixed point iteration: \( x_{k+1} = g(x_k) \) is locally linearly convergent if for \( x^* = g(x^*) \), we have \( |g'(x^*)| < 1 \):
  \[
e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*) = g(x^*) + g'(x^*)(x^* - x_k) + ... - g(x^*)
\]

- It is quadratically convergent if \( g'(x^*) = 0 \):
  \[
e_{k+1} = g''(x^*)(x^* - x_k)^2 / 2 + O((x^* - x_k)^3)
\]
Newton’s Method

- Newton’s method is derived from a *Taylor series* expansion of $f$ at $x_k$:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

- Newton’s method is *quadratically convergent* if started sufficiently close to $x^*$ so long as $f'(x^*) \neq 0$: 

The Secant method approximates $f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$.

The convergence is superlinear but not quadratic:
Nonlinear Tangential Interpolants

- Secant method uses a linear interpolant based on points $f(x_k), f(x_{k-1})$, could use more points and higher-order interpolant:

- Quadratic interpolation (Muller’s method) achieves convergence rate $r \approx 1.84$: 
Solving Nonlinear Equations Method Summary

- Newton’s and secant method provide basic approaches for solving a univariate nonlinear equation:

- Inverse (quadratic) interpolation can provide better convergence:
Achieving Global Convergence

- Hybrid bisection/Newton methods:
  - Bounded (damped) step-size:
Systems of Nonlinear Equations

- Given \( f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} \) for \( x \in \mathbb{R}^n \), seek \( x^* \in \mathbb{R}^n \) so that \( f(x^*) = 0 \)

- At a particular point \( x \), the *Jacobian* of \( f \), describes how \( f \) changes in a given direction of change in \( x \),

\[
J_f(x) = \begin{bmatrix}
\frac{df_1}{dx_1}(x) & \cdots & \frac{df_1}{dx_n}(x) \\
\vdots & \ddots & \vdots \\
\frac{df_m}{dx_1}(x) & \cdots & \frac{df_m}{dx_n}(x)
\end{bmatrix}
\]
Multivariate Newton Iteration

- Fixed-point iteration \( x_{k+1} = g(x_k) \) achieves local convergence so long as 
  \[ |\lambda_{\text{max}}(J_g(x^*))| < 1. \]

- Newton’s method corresponds to the fixed-point iteration

\[
g(x) = x - J_f^{-1}(x)f(x)
\]
Convergence of Newton Iteration

- Newton’s method achieves quadratic local convergence if \( \| J_f^{-1}(x^*) \| \) is bounded:
Convergence of Newton Iteration (II)

- Quadratic convergence is achieved when the Jacobian of a fixed-point iteration is zero at the solution, which is true for Newton’s method:
Estimating the Jacobian using Finite Differences

- To obtain $J_f(x_k)$ at iteration $k$, can use finite differences:

- $n + 1$ function evaluations are needed: $f(x), f(x + he_i) \forall i \in \{1, \ldots, n\}$, which correspond to $m(n + 1)$ scalar function evaluations.
Cost of Multivariate Newton Iteration

- What is the cost of solving $J_f(x_k)s_k = f(x_k)$?

- What is the cost of Newton’s iteration overall?
Secant Updating Methods

In solving a nonlinear equation, seek approximate Jacobian $J_f(x_k)$ for each $x_k$

- Find $B_{k+1} = B_k + \delta B_k \approx J_f(x_{k+1})$, so as to approximate secant equation

$$B_{k+1}(x_{k+1} - x_k) = \underbrace{f(x_{k+1}) - f(x_k)}_{\delta f} = \underbrace{\delta x}_{\delta x}$$

- Broyden’s method is given by minimizing $||\delta B_k||_F$:

$$\delta B_k = \frac{\delta f - B_k \delta x}{||\delta x||^2} \delta x^T$$
Newton-Like Methods

- Can dampen step-size to improve reliability of Newton or Broyden iteration:

  - *Trust region methods* provide general step-size control: