CS 450: Numerical Anlaysis[1]

Nonlinear Equations

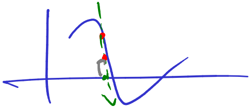
University of Illinois at Urbana-Champaign

[1] *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
  - $f(\boldsymbol{x}^*) = 0$ multi-variate, scalar valued
  - $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$, vector-valued $f\left(\begin{bmatrix} \ \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
  - 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 $\boldsymbol{f}$:

Newton's method



$f(x) = 0$

$f(x_k + \delta x) = f(x_k) + \delta x f'(x_k) = 0$

current guess

$\delta x = -f(x_k)/f'(x_k)$

$x_{k+1} = x_k + \delta x = x_k - f(x_k)/f'(x_k)$

Jacobian $\quad J_f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & \ddots & \\ \frac{\partial f_n}{\partial x_1} & & \end{bmatrix} \qquad f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix}$

$x \in R^m \qquad J_f(x) \in R^{n \times m}$

$f(x) \in R^n \qquad f(x^{(k)} + \delta x) \approx f(x^{(k)}) + J_f(x^{(k)}) \delta x = 0$

$m = n \qquad J_f(x^{(k)}) \delta x = -f(x^{(k)})$

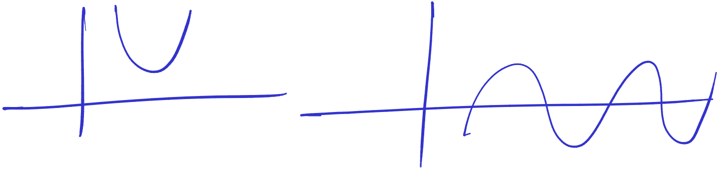
$$\delta x = -J_f(x^{(k)})^{-1} f(x^{(k)})$$

$$x^{(k+1)} = x^{(k)} - J_f(x^{(k)})^{-1} f(x^{(k)})$$

Demo: *Three quadratic functions*

Nonexistence and Nonuniqueness of Solutions

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



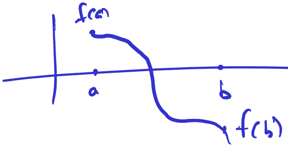
- Solutions in the multivariate case correspond to intersections of hypersurfaces:

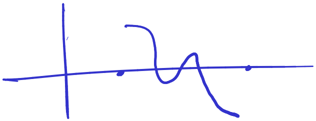
curves

Conditions for Existence of Solution

- *Intermediate value theorem* for univariate problems:

bracket $[a, b]$ $\quad$ $\text{sign}(f(a)) \neq \text{sign}(f(b))$





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

$$g(x) = f(x) + x$$

$$g(x^*) = x^*$$

$$\Rightarrow f(x^*) = 0$$

$$\|g(x) - g(z)\| \leq \gamma \|x - z\|$$

Lipschitz continuity

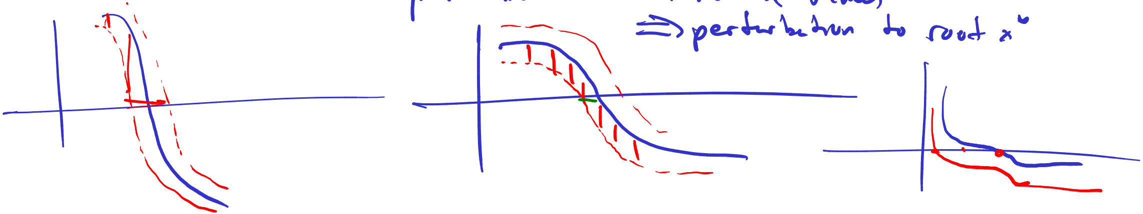
Lipschitz constant

Conditioning of Nonlinear Equations

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

perturbation to function values

$\Rightarrow$ perturbation to root x^*



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

κ_{abs}(root finding for f for root x^*) is $1/f'(x^*)$

Multiple Roots and Degeneracy

root is simple if $m=1$

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

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

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

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

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

- Increased multiplicity affects conditioning and convergence:

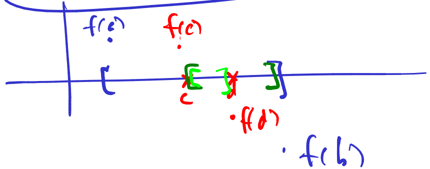
$m>1 \implies$ ill-posed

can modify problem/algorithm to handle $m>1$

***Demo:** Bisection Method*

Bisection Algorithm

- Assume we know the desired root exists in a bracket $[a, b]$ and $sign(f(a)) \neq sign(f(b))$.



invariant:

- Bisection subdivides the interval by a factor of two at each step by considering $f(c_k)$ at $c_k = (a_k + b_k)/2$:

Rates of Convergence

- Let x_k be the kth iterate and $e_k = x_k = x^*$ be the error, bisection obtains *linear convergence*, $\lim_{k\to\infty} ||e_k||/||e_{k-1}|| \leq C$:

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

for bisection

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

$e_k \leq \varepsilon$, e.g. $\varepsilon = .0001$

$O(\log(1/\varepsilon))$ digits of accuracy

- rth 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

$O(\log_r(\log(1/\varepsilon)))$ steps

$O(\log(\log(1/\varepsilon)))$

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 $|g'(x^*)| < 1$:

$x_k \to x^*$

$$e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*)$$

$$= \cancel{g(x^*)} + g'(x^*)(x^* - x_k) + \dots \quad \cancel{- g(x^*)}$$

Taylor-exp of g at x^*

$$= g'(x^*)\underbrace{(x^* - x_k)}_{e_k} + O((x^* - x_k)^2)$$

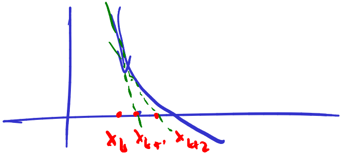
- It is quadratically convergent if $g'(x^*) = 0$:

$$e_{k+1} = g''(x^*)\underbrace{(x^* - x_k)}_{e_k}^2/2 + O((x^* - x_k)^3)$$

Demo: *Newton's Method*
Demo: *Convergence of Newton's Method*

Newton's Method

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



$$x_{k+1} = \underbrace{x_k - f(x_k) / f'(x_k)}_{g(x_k)}$$

fixed-point function

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

Secant Method

Demo: *Secant Method*
Demo: *Convergence of the Secant Method*

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

- At a particular point $\boldsymbol{x}$, the *Jacobian* of $\boldsymbol{f}$, describes how $\boldsymbol{f}$ changes in a given direction of change in $\boldsymbol{x}$,

$$\boldsymbol{J_f}(\boldsymbol{x}) = \begin{bmatrix} \frac{df_1}{dx_1}(\boldsymbol{x}) & \cdots & \frac{df_1}{dx_n}(\boldsymbol{x}) \\ \vdots & & \vdots \\ \frac{df_m}{dx_1}(\boldsymbol{x}) & \cdots & \frac{df_m}{dx_n}(\boldsymbol{x}) \end{bmatrix}$$

Multivariate Newton Iteration

- Fixed-point iteration $\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k)$ achieves local convergence so long as $|\lambda_{\text{max}}(\boldsymbol{J_g}(\boldsymbol{x}^*))| < 1$:

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

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{J}_{\boldsymbol{f}}^{-1}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$$

Convergence of Newton Iteration

- Newton's method achieves quadratic local convergence if $||\boldsymbol{J}_{\boldsymbol{f}}^{-1}(\boldsymbol{x}^*)||$ 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 $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}_k)$ at iteration k, can use finite differences:

- $n+1$ function evaluations are needed: $\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}+h\boldsymbol{e}_i)\forall i \in \{1,\ldots,n\}$, which correspond to $m(n+1)$ scalar function evaluations.

Cost of Multivariate Newton Iteration

- What is the cost of solving $\boldsymbol{J_f}(\boldsymbol{x}_k)\boldsymbol{s}_k = \boldsymbol{f}(\boldsymbol{x}_k)$?

- What is the cost of Newton's iteration overall?

Secant Updating Methods

In solving a nonlinear equation, seek approximate Jacobian $\boldsymbol{J_f}(\boldsymbol{x}_k)$ for each $\boldsymbol{x}_k$

- Find $\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \boldsymbol{\delta B}_k \approx \boldsymbol{J_f}(\boldsymbol{x}_{k+1})$, so as to approximate *secant equation*

$$\boldsymbol{B}_{k+1}\underbrace{(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)}_{\boldsymbol{\delta x}} = \underbrace{\boldsymbol{f}(\boldsymbol{x}_{k+1}) - \boldsymbol{f}(\boldsymbol{x}_k)}_{\boldsymbol{\delta f}}$$

- *Broyden's method* is given by minimizing $||\boldsymbol{\delta B}_k||_F$:

$$\boldsymbol{\delta B}_k = \frac{\boldsymbol{\delta f} - \boldsymbol{B}_k\boldsymbol{\delta x}}{||\boldsymbol{\delta x}||^2}\boldsymbol{\delta x}^T$$

Newton-Like Methods

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

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