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:

- Solving nonlinear equations has many applications:

Solving Nonlinear Equations

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

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:

Conditions for Existence of Solution

- *Intermediate value theorem* for univariate problems:

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

$$||\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{z})|| \leq \gamma ||\boldsymbol{x} - \boldsymbol{z}||, 0 \leq \gamma < 1$$

Conditioning of Nonlinear Equations

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

- 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 $||\boldsymbol{J}_{\boldsymbol{f}}^{-1}(\boldsymbol{x}^*)||$:

Multiple Roots and Degeneracy

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

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

- Increased multiplicity affects conditioning and convergence:

Bisection Algorithm

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

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

Convergence of Fixed Point Iteration

- Fixed point iteration: $x_{k+1} = g(x_k)$ is locally linearly convergent to fixed point x^* if g is continuously differentiable near x^* and $|g'(x^*)| < 1$:

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

Newton's Method

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

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

- Newton's method is *quadratically convergent* if started sufficiently close to x^* so long as $f'(x^*) \neq 0$ and f is twice continuously differentiable in the neighborhood of x^*:

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 of the Secant method 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) can achieve a convergence order of $r \approx 1.84$:
- Inverse quadratic interpolation resolves the problem of nonexistence/nonuniqueness of roots of polynomial interpolants:

Achieving Global Convergence

- Hybrid bisection/Newton methods:

- Bounded (damped) step-size:

Systems of Nonlinear Equations

- Given $\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) & \cdots & f_m(\boldsymbol{x}) \end{bmatrix}^T$ for $\boldsymbol{x} \in \mathbb{R}^n$, seek $\boldsymbol{x}^*$ 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 $\boldsymbol{x}$,

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

Multivariate Newton Iteration

- Fixed-point iteration $\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k)$ achieves local convergence if (in addition to contraints on differentiability of $\boldsymbol{g}$) we have $|\lambda_{\text{max}}(\boldsymbol{J_g}(\boldsymbol{x}^*))| < 1$ and quadratic convergence if $\boldsymbol{J_g}(\boldsymbol{x}^*) = \boldsymbol{O}$:

Multidimensional Newton's Method

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

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

- 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 $\boldsymbol{J_f}(\boldsymbol{x}_k)$ at iteration k, can use finite differences:

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

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?

Quasi-Newton 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* solves the secant equation and minimizes $||\boldsymbol{\delta B}_k||_F$:

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

Safeguarding Methods

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

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