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:
  - $f(x^*) = 0$ *univariate nonlinear function*
  - $f(\boldsymbol{x}^*) = 0$ *multivariate, scalar-valued nonlinear function*
  - $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$ *multivariate, vector-valued nonlinear function*
  - 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 for univariate functions starting at point x_k finds root of* $h(\delta x) = f(x_k) + f'(x_k)\delta x \approx f(x_k + \delta x)$*, so*

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

  - *Newton's method for multivariate functions starting at point $\boldsymbol{x}_k$ finds root of* $\boldsymbol{h}(\boldsymbol{\delta x}) = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{J_f}(\boldsymbol{x}_k)\boldsymbol{\delta x} \approx \boldsymbol{f}(\boldsymbol{x}_k + \boldsymbol{\delta x})$*, with Jacobian* $(\boldsymbol{J_f}(\boldsymbol{x}))_{ij} = \frac{\delta f_i}{\delta x_j}(\boldsymbol{x})$*, so*

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

Demo: *Three quadratic functions*

Nonexistence and Nonuniqueness of Solutions

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

  Consider functions that are strictly greater than zero or have many zeros.

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

  The zeros of each equation define a hypersurface in $\mathbb{R}^n$, in the linear case, these are hyperplanes. Intersections of hypersurfaces for many equations, define the solutions, which are roots of all equations.

  Consider that two curves can intersect at many points in space. Two hypersurfaces in three-dimensional space may not intersect or may have multiple curves of intersection.

Conditions for Existence of Solution

- *Intermediate value theorem* for univariate problems:

  If for $a < b$, $sign(f(a)) \neq sign(f(b))$ and f is continuous, then $[a, b]$ is a bracket that contains a root,

  $$\exists x^* \in [a, b], \quad f(x^*) = 0.$$

- 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,

  $$||\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{z})|| \leq \gamma ||\boldsymbol{x} - \boldsymbol{z}||$$

  - *Contained implies that in the domain S, for any $\boldsymbol{x} \in S$, $\boldsymbol{g}(\boldsymbol{x}) \in S$, while contractive implies that the function is Lipschitz continuous in S.*
  - *When solving for a root of $\boldsymbol{f}$, can define various fixed point functions $\boldsymbol{g}$, so that their solution $\boldsymbol{g}(\boldsymbol{x}^*) = \boldsymbol{x}^*$ provides a root of $\boldsymbol{f}$, $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$, the simplest being $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{x}$.*

Conditioning of Nonlinear Equations

- Generally, we take interest in the absolute rather than relative conditioning of solving $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$:
  - *The sensitivity of solving a nonlinear equation is the ratio of magnitudes of the perturbation to the root and perturbation to the function values.*
  - *It makes sense to consider absolute perturbations to $\boldsymbol{f}$, since any nonzero perturbation to function values is infinite in magnitude relative to $\boldsymbol{f}(\boldsymbol{x}^*) = \mathbf{0}$.*
- 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 $||\boldsymbol{J}_{\boldsymbol{f}}^{-1}(\boldsymbol{x}^*)||$:
  - *If we change the value of f by at most δf at any point in the function while maintaining continuity, the root will shift by at most $|\delta f|/|f'(x^*)|$ assuming $|\delta f|$ is sufficiently small.*
  - *This relationship is the converse of conditioning in function evalution, where a perturbation to input x, results in a perturbation of at most $\kappa_{abs}(f) = |f'(x)|$ larger to the function value.*

Multiple Roots and Degeneracy

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

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

Proof: for some function $t^{(0)}(x)$, we have that

$$\begin{aligned} f(x) &= (x-x^*)^m t^{(0)}(x), \\ f'(x) &= m(x-x^*)^{m-1} t^{(0)}(x) + (x-x^*)^m t^{(0)\prime}(x) \\ &\equiv (x-x^*)^{m-1} t^{(1)}(x), \\ f^{(m-1)}(x) &= (x-x^*) t^{(m-1)}(x), \end{aligned}$$

where $t^{(i)} = (m-i+1)t^{(i-1)}(x) - (x-x^*)t^{(i-1)\prime}(x)$.

- Increased multiplicity affects conditioning and convergence:
  - *When a root x^* not simple, i.e. $m>1$, then $f'(x^*)=0$, so the problem of finding that root is ill-posed as $1/|f'(x^*)| = \infty$.*
  - *In practice, this means we have multiple roots at the same x^* which are impossible to distinguish and may need to reformulate problem/algorithms.*

Bisection Algorithm

- Assume we know the desired root exists in a bracket $[a, b]$ and $sign(f(a)) \neq sign(f(b))$:
  - *if f is continuous, by the inverse value theorem, the bracket contains a root*
  - *can proceed to narrow interval to find a root*
  - *one caveat is that multiple roots may exist in $[a, b]$*
  - *another caveat is that we've imposed a restrictive condition, it can be difficult to find two points where the function has opposite sign*
- Bisection subdivides the interval by a factor of two at each step by considering $f(c_k)$ at $c_k = (a_k + b_k)/2$:

$$[a_{k+1}, b_{k+1}] = \begin{cases} [c_k, b_k] & : sign(f(a_k)) = sign(f(c_k)) \\ [a_k, c_k] & : sign(f(b_k)) = sign(f(c_k)) \end{cases}$$

Rates of Convergence

- Let $\boldsymbol{x}_k$ be the kth iterate and $\boldsymbol{e}_k = \boldsymbol{x}_k = \boldsymbol{x}^*$ be the error, bisection obtains *linear convergence*, $\lim_{k\to\infty} ||\boldsymbol{e}_k||/||\boldsymbol{e}_{k-1}|| \le C$:

  In bisection, working with the natural error bound given by bracket size,

  $$e_k = b_k - a_k = \frac{1}{2}(b_{k-1} - a_{k-1}) = \frac{1}{2}e_{k-1},$$

  so bisection achieves linear convergence with $C = 1/2$. *With linear convergence, error* $e_k \le \epsilon$ *is achieved after* $O(\log_C(1/\epsilon))$ *steps.*

- rth order convergence implies that $||\boldsymbol{e}_k||/||\boldsymbol{e}_{k-1}||^r \le C$
  - *Convergence of order* $r > 1$ *(superlinear) implies that the number of digits of correctness increases by a factor of* r *at each step.*
  - *For* $r > 1$, *error* $e_k \le \epsilon$ *is achieved after* $O(\log_r(\log(1/\epsilon)))$ *steps.*
  - *Having achieved superlinear convergence* ($r > 1$), *methods differ only by constant factors in complexity.*

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

  Note that,

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

  By taking the Taylor expansion of g at x^* to represent $g(x_k)$, we can observe

  $$\begin{aligned} e_{k+1} &= g'(x^*)(x_k - x^*) + O((x_k - x^*)^2) \\ &= g'(x^*)e_k + O((x_k - x^*)^2), \end{aligned}$$

  where the asymptotic term decreases as x_k approaches x^*.

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

  Taking the same Taylor expansion, the leading term is now zero and we obtain

  $$\begin{aligned} e_{k+1} &= g''(x^*)(x_k - x^*)^2/2 + O((x_k - x^*)^3) \\ &= g''(x^*)e_k^2/2 + O((x_k - x^*)^3). \end{aligned}$$

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:

$$f(x) = \underbrace{f(x_k) + f'(x_k)(x - x_k)}_{\textit{secant line approximation}} + (1/2!)f''(x_k)(x - x_k)^2 + \cdots$$

- Newton's method is *quadratically convergent* if started sufficiently close to x^* so long as $f'(x^*) \neq 0$:
  - *Newton's method corresponds to the fixed point function $g(x) = x - f(x)/f'(x)$.*
  - *It achieves quadratic convergence since $g'(x^*) = 0$,*

$$g'(x^*) = 1 - \underbrace{f'(x^*)/f'(x^*)}_{1} + \underbrace{f(x^*)}_{0} f''(x^*)/f'(x^*)^2.$$

Secant Method

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

- The *Secant method* approximates $f'(x_k) \approx \frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$:
  - *Usually, this method is the cheapest approximation possible, since function values $f(x_k)$ and $f(x_{k-1})$ are already available.*
  - *Approximation quality depends on magnitude $f(x_k) - f(x_{k-1})$ and $x_k - x_{k-1}$.*
  - *If the two points are far apart, the derivative approximation may be bad locally, while if they are very close we have to take care in handling cancellation.*
  - *A well-chosen finite-difference step at each x_k provides a more robust approximation, but requires another function evaluation.*
- The convergence of the Secant method is *superlinear* but not quadratic:
  - *The error will now depend on the previous two errors, since we are using the previous two points.*
  - *In simplified form, the error at the kth iteration satisfies $e_k \le e_{k-1}e_{k-2}$.*
  - *Note that $\log(e_k) = \log(e_{k-1}) + \log(e_{k-2})$ is the Fibonacci sequence, which grows at a rate of $r = (1+\sqrt{5})/2$.*
  - *Thus the (negative) exponent of the error increases by roughly a factor of r at each step, i.e. the convergence rate is r.*

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:

  Have points $(x_0, f(x_0)), \dots, (x_k, f(x_k))$ *can fit polynomial to* $p(x_i) = f(x_i)$ *for some subset of points* $x_i \in S \subseteq \{x_0, \dots x_k\}$*.*

- Quadratic interpolation (Muller's method) achieves convergence rate $r \approx 1.84$:

  Quadratic interpolation requires three points x_{k-2}*,* x_{k-1}*, and* x_k*.*

- Inverse quadratic interpolation resolves the problem of nonexistence/nonuniqueness of roots of polynomial interpolants:
  - *Interpolate quadratic polynomial* q *so that* $q(f(x_i)) = x_i$ *for* $i \in \{k, k-1, k-2\}$*.*
  - *Approximate root simply computed by evaluating interpolant at zero* $x_{k+1} = q(0)$*.*

Achieving Global Convergence

- Hybrid bisection/Newton methods:

  Given a bracket (interval), can proceed with bisection until bracket is small then switch to Newton. Alternatively, can attempt Newton, check if it stays within bracket (safeguard) and proceed with change only if it does.

- Bounded (damped) step-size:

  Newton's method gives us a direction. Decreasing the step size in that direction trades off convergence rate for reliability. We will study how step sizes can be chosen in more detail in the context of optimization.

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 $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$
  - *$\boldsymbol{x}^*$ must simultaneously set to zero all components of $\boldsymbol{f}$: $f_1(\boldsymbol{x}^*) = \cdots = f_m(\boldsymbol{x}^*) = 0$.*
  - *We focus on the case of $m = n$, so that the number of equations matches the number of unknowns.*

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

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

  Our local linear approximation is given by

$$\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{\delta x}) \approx \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{J_f}(\boldsymbol{x})\boldsymbol{\delta x},$$

  note that when $m = 1$ the Jacobian corresponds to the gradient of $\boldsymbol{f}$.

Multivariate Newton Iteration

- Fixed-point iteration $\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k)$ achieves local convergence so long as $|\lambda_{\max}(\boldsymbol{J_g}(\boldsymbol{x}^*))| < 1$ and quadratic convergence if $\boldsymbol{J_g} = \boldsymbol{O}$:
  Taking a multivariate Taylor expansion of $\boldsymbol{g}(\boldsymbol{x}_k)$ at center $\boldsymbol{x}^*$ we get

$$
\begin{aligned}
\boldsymbol{e}_k &= \boldsymbol{x}^* - \boldsymbol{x}_{k+1} \\
&= \boldsymbol{g}(\boldsymbol{x}^*) - \boldsymbol{g}(\boldsymbol{x}_k) \\
&= \underbrace{\boldsymbol{J_g}(\boldsymbol{x}^*)}_{\boldsymbol{O}}(\boldsymbol{x}^* - \boldsymbol{x}_k) \\
&\quad + \frac{1}{2}\begin{bmatrix}\boldsymbol{H}_{\boldsymbol{g}}^{(1)}(\boldsymbol{x}^*)\cdot(\boldsymbol{x}^* - \boldsymbol{x}_k) & \cdots & \boldsymbol{H}_{\boldsymbol{g}}^{(n)}(\boldsymbol{x}^*)\cdot(\boldsymbol{x}^* - \boldsymbol{x}_k)\end{bmatrix}(\boldsymbol{x}^* - \boldsymbol{x}_k) \\
||\boldsymbol{e}_k|| &= O(\max_i ||\boldsymbol{H}_{\boldsymbol{g}}^{(i)}(\boldsymbol{x}^*)|| \cdot ||\underbrace{\boldsymbol{x}^* - \boldsymbol{x}_k}_{\boldsymbol{e}_k}||^2) \\
&= O(||\boldsymbol{e}_k||^2)
\end{aligned}
$$

where $\boldsymbol{H}_{\boldsymbol{g}}^{(i)}$ is the ith component of the derivative of $\boldsymbol{J_g}(\boldsymbol{x}^*)$.

Multidimensional Newton's Method

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

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

  - *Note that generally Newton's method iteration has a fixed-point $\bar{\boldsymbol{x}}$ whenever $\boldsymbol{f}(\bar{\boldsymbol{x}}) = \boldsymbol{0}$, i.e. we have found a root of $\boldsymbol{f}$, namely $\boldsymbol{x}^* = \bar{\boldsymbol{x}}$.*
  - *Generally, we require that $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}^*)$ is nonsingular, otherwise we can find nonzero solutions $\boldsymbol{y}$ to $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}^*)\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$.*

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

$$\begin{aligned}
\boldsymbol{g}(\boldsymbol{x}^*) &= \boldsymbol{x}^* - \boldsymbol{J}_{\boldsymbol{f}}^{-1}(\boldsymbol{x}^*)\boldsymbol{f}(\boldsymbol{x}^*) \\
\boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}^*) &= \boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{f}}^{-1}(\boldsymbol{x}^*)\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}^*) - \sum_i \underbrace{f_i(\boldsymbol{x}^*)}_{0}\boldsymbol{H}_{\boldsymbol{f}}^{(i)}(\boldsymbol{x}^*) \\
&= \boldsymbol{I} - \boldsymbol{I} - \boldsymbol{O} = \boldsymbol{O}
\end{aligned}$$

where $\boldsymbol{H}_{\boldsymbol{f}}^{(i)}$ is the ith component of the derivative of $\boldsymbol{J}_{\boldsymbol{f}}^{-1}(\boldsymbol{x}^*)$.

Estimating the Jacobian using Finite Differences

- To obtain $\boldsymbol{J_f}(\boldsymbol{x}_k)$ at iteration k, can use finite differences:
  - *If $\boldsymbol{J_f}(x) \in \mathbb{R}^{m\times 1}$ (single-variate but vector-valued $\boldsymbol{f}$), we can estimate*

$$\boldsymbol{J_f}(x_k) \approx (1/h)(\boldsymbol{f}(x_k + h) - \boldsymbol{f}(x_k)).$$

  - *More generally, the ith column of $\boldsymbol{j}_i$ of the Jacobian $\boldsymbol{J_f}(\boldsymbol{x}_k)$ can be estimated by*

$$\boldsymbol{j}_i \approx (1/h)(\boldsymbol{f}(\boldsymbol{x}_k + h\boldsymbol{e}_i) - \boldsymbol{f}(\boldsymbol{x}_k)).$$

- $n+1$ 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 $\boldsymbol{J_f}(\boldsymbol{x}_k)\boldsymbol{s}_k = \boldsymbol{f}(\boldsymbol{x}_k)$?
  $O(n^3)$
- What is the cost of Newton's iteration overall?
  For k steps, $O(n^3k + kn^2\gamma_{\textit{func-eval}})$.

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*

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

  Generally, the secant equation is underdetermined, as we usually need n finite-difference formulas to determine $\boldsymbol{J_f}(\boldsymbol{x}_k)$, so the secant updating methods find only approximate $\boldsymbol{B}_{k+1}$, usually as a modification of $\boldsymbol{B}_k$.

- *Broyden's method* solves the secant equation and minimizes $||\boldsymbol{\delta B}_k||_F$:

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

  Note that $\boldsymbol{\delta B}_k$ is rank-1. Consequently, we can use the Sherman-Morrison formula to update $\boldsymbol{B}_{k+1}^{-1}$ with $O(n^2)$ work. Various other variants exist.

Safeguarding Methods

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

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k \quad \textit{where} \quad \alpha_k \leq 1$$

  can pick α_k *so to ensure* $||\boldsymbol{f}(\boldsymbol{x}_{k+1})|| < ||\boldsymbol{f}(\boldsymbol{x}_k)||$ *or by doing a line-search to minimize* $||\boldsymbol{f}(\boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k)||$.

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

  Establish/maintain/update region within which step is expected to be accurate. Pick each step to stay within trust region while minimizing $||\boldsymbol{f}(\boldsymbol{x}_{k+1})||$*. Observe that the Newton-like generally seek to progress to a minima of* $||\boldsymbol{f}(\boldsymbol{x}_{k+1})||$*, and indeed much of the theory of these methods targets optimization.*