Lecture 06: Rootfinding and Optimization for functions of a single variable f(x)

Outline

1) Fail-safe hybrid methods
   NewtSafe (Newton + Bisection: Numerical Recipes)
   Brent's method (Secant+IQI+Bisection)
   Matlab's fzero (Brent's method)
2) Examples and Demo's
3) Optimization algorithms to find min(f(x)) on [a,b]
   Bracketing Algorithms: Golden Section Search
   Interpolation Algorithms: Successive Parabolic Interpolation
   Hybrid Methods: fminbnd

Hybrid Methods:

Design Goals:

1) Robustness: Given a bracket [a,b], maintain the bracket
2) Efficiency: Use superlinear convergent methods when possible

Some Options:

Have derivatives:
   NewtSafe (or RootSafe, Numerical Recipes)
   Newton's method within a bracket, Bisection otherwise

No Derivatives:
   Brent's Algorithm (zbrent Numerical Recipes, fzero Matlab)
   Returns minimum bracket using combination of
   Bisection
   Secant method
   Inverse Quadratic Interpolation
NewtSafe Algorithm:

1. Initialize \( f(x) \) \((a,b)\), \( C = a \) \( f_c = f(c) \)
2. \( f_a = f(a) \), \( f_b = f(b) \) \( \frac{df}{dx} = f(c) \) \( h = b - a \)
3. Until converged
   - Try a Newton step \( C = C - \frac{f_c}{df} \)
   - If \( C \notin (a,b) \)
     - \( C = a + \frac{b - a}{2} \)
     - \( [f_c, df] = f(c) \)
   - If \( \text{sign}(f_b) \neq \text{sign}(f_c) \)
     - \( a = c \) \( f_a = f_c \)
   - Else
     - \( b = c \) \( f_b = f_c \)
     - \( h = b - a \) or \( |f_c| < \text{tol} \)
     - If \( (|h| < b \times \text{tol}) \) \( \text{break} \)

Example: \( f(x) = \sin(2\pi x) \)
**Brent-Dekker Algorithm:**

*Hybrid method using IQI+Secant+Bisection (foolproof)*

**Given:** \(f(x)\) and a bracket \([a,b]\)

**Initialize:** use Secant method to find \(c\) between \(a\) and \(b\)

**Until Converged:** \(|b-a|<\text{tol}b\) or \(f(c)=0\)

Arrange \(a\), \(b\) and \(c\) so that
- \(a\) and \(b\) form a bracket
- \(|f(b)| <= |f(a)|\)
- \(c\) is the previous value of \(b\)

if \(c != a:\)
    Try \(c=\text{IQI}\)
else \(c=a\)
    Try \(c=\text{Secant}\)
end

if \(c\) in the bracket
    keep it
else
    use \(c=\text{Bisection}\)
end
Brent's method: Bisection + Secant + IQI
The Geometric picture
Example: \( f(x) = \sin(2\pi x) \)

Brent-Dekker Algorithm:

Comments:

This algorithm is Bullet-proof

It's guaranteed to always maintain a bracket

Doesn't require derivatives

Uses rapidly converging methods when reliable

Uses slow but sure method when necessary

In Matlab: fzero...demonstrate with fzerogui (code in fzerotxt)

basic syntax (see help fzero, help optimset)

options = optimset(['disp','iter'])

x = fzero(@func,x0,options)

if x0 is scalar, it searches for a bracket if x0 = [a b] it tests for a bracket and fails if sign(f(a)) ≠ sign(f(b))
Optimization (finding extrema) for functions of one variable

Closely related problem to root finding, but rather than finding \( f(x) = 0 \) on some interval. *Find \( \min(f(x)) \) on some interval...*

---

Optimization (finding extrema) for functions of one variable

General algorithms (similar to rootfinding algorithms):

- **Bracketing algorithms:** Golden-Section Search (linear convergence)
- **Interpolation algorithms:** repeated parabolic interpolation
- **Hybrid algorithms:** Matlab’s fminbnd(func,a,b,tol)
Bracketing Algorithm: Golden Section Search

Like Bisection: given f(x) in C[a,b] that is convex (uni-modal) over an interval [a,b]
reduce the interval size until it “brackets” the minimum

Note: bracketing not as well defined, you should always plot your function

Questions:
1) How many points are required to approximate a minimum?
2) How many points are needed to subdivide? [a,b]?
3) How to choose those points efficiently
Golden Section Search: The Algorithm

Given f(x) and unimodal bracket [a, b]

\[ a \quad \rho \quad 1-\rho \quad b \]
\[ \quad u \quad v \]

Initialize:

- \( f_a = f(a) \)
- \( f_b = f(b) \)
- \( u = a + \rho (b-a) \)
- \( v = a + (1-\rho) b-a \)

Loop until \( |h| < \text{tol} \)

- \( f_u = f(u) \)
- \( f_v = f(v) \)

If \( f_u < f_v \)

- \( b = v \)
- \( v = u \)
- \( u = a + \rho (b-a) \)
- \( f_v = f(u) \)

Else

- \( a = u \)
- \( u = v \)
- \( f_u = f(v) \)
- \( h = (3-\sqrt{5})/2 \approx 0.382 \)
- \( v = a + \rho (b-a) \)
- \( u = a + (1-\rho) h \)
- \( f_v = f(v) \)

end

Golden Section Search: The corrected and cleaned up Algorithm

Given f(x) and unimodal bracket [a, b]

\[ a \quad \rho \quad 1-\rho \quad b \]
\[ \quad u \quad v \]

Initialize:

- \( f_a = f(a) \)
- \( f_b = f(b) \)
- \( h = b-a \)
- \( \rho = 2-\phi = (3-\sqrt{5})/2 \approx 0.382 \)
- \( u = a + \rho h \)
- \( v = a + (1-\rho) h \)

while \( |h| > \text{tol} \)

- if \( f_u < f_v \) % minimum is between a and v
  - \( b = v \)
  - \( f_b = f(v) \)
  - \( v = u \)
  - \( f_v = f(u) \)
  - \( h = b-a \)
  - \( u = a + \rho h \)
  - \( f_u = f(u) \)

else % minimum is between u and b

- \( a = u \)
- \( f_a = f(u) \)
- \( u = v \)
- \( f_u = f(v) \)
- \( h = b-a \)
- \( v = a + (1-\rho) h \)
- \( f_v = f(v) \)

end
Interpolation Algorithm: Successive Parabolic Interpolation

Like Secant: use multiple samplings of the function to approximate the function.

Successive Parabolic Interpolation: The Algorithm

Given: \( f(x) \) and \([a, b]\)

initialize: \( x = [a \ b \ (a+b)/2] \)
\( n = 2: -1: 0; \)

for \( i = 1: \text{MAX\_ITERATIONS} \)
\( f = \text{func}(x); \)
\( p = \text{polyfit}(x, f, 2); \)
\( pPrime = n.*p; \)
\( xNew(i) = -pPrime(2)/pPrime(1); \)
\( x = [x(2:end) \ xNew(i)]; \)
\( \text{relErr} = \text{abs}(xNew(i)-xNew(i-1))/\text{abs}(xNew(i)); \)
if \( \text{relErr} < \text{tol} \)
\( \text{break} \)
end
end
Hybrid schemes: *fminbnd*
successive Parabolic interpolation + golden section search

in *Matlab*: use *fminbnd*

syntax:

```plaintext
options = optimset('disp','iter')
x = fminbnd(@func,x1,x2,options)
```

examples:

```plaintext
x = fminbnd(@(x) sin(2*pi*x),0,1,options)
x = fminbnd(@(x) -humps(x),.1,.4,options)
```