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: Bisection + Newton
The Geometric picture
Example: f(x)=\sin(2\pi x)

\[ f(x) = \sin(2\pi x) \]

NewtSafe Algorithm:
NewtSafe Code:

```matlab
function [xFinal, xN, errorN] = newtSafe(func,a,b,tol)
% NEWTSAFE - root-finder using hybrid Newton-Bisection method to always maintain bracket
%   [xFinal, xN, errorN] = newtSafe(func,a,b,tol)
%    func:  function [f, df] =func(x); returns both the function and its derivative
%    a,b:     initial bracket
%    tol:    stopping condition for  error f(x) <= tol or |b-a|<tol*b
%    xFinal:   final value
%    xN:       vector of intermediate iterates
%    errorN:   vector of errors

MAX_ITERATIONS = 100;
VERBOSE = true;
% initialize the problem
h = b-a;
fa = func(a);
fb = func(b);
if ( sign(fa) == sign(fb) )
    error('function must be bracketed to begin with');
end
 c = a;
% start on the left side (could also choose the middle
[ fc, df] = func(c);
xN(1) = c;
errorN(1,:) = [ abs(fc) h ];
%  begin iteration until convergence or Maximum Iterations
for i = 1:MAX_ITERATIONS
% try a Newton step
useNewton = true;
c = c - fc/df;
% if not in bracket choose bisection
if ( ~(a <= c & b >= c) )
c = a + h/2;
useNewton = false;
end
% Evaluate function and derivative at new c
[fc,df]=func(c);
% check and maintain bracket
if ( sign(fc) ~= sign(fb) )
a=c;
fa=fc;
else
b = c;
fb = fc;
end
h = b-a;
% calculate errors and track solutions
absError = abs(fc);
relError = h;
xN(i) = c;
errorN(i,:) = [ absError, relError ];
% be yacky
if VERBOSE
if useNewton
    disp(sprintf('i=%d  Newton: a=%12.8e, b=%12.8e, c=%12.8e, f(c)=%12.8e, h=%12.8e',i,a,b,c,fc,h));
else
    disp(sprintf('i=%d  Bisect   : a=%12.8e, b=%12.8e, c=%12.8e, f(c)=%12.8e, h=%12.8e',i,a,b,c,fc,h));
end
end
% check if converged
if ( absError < tol || relError < tol*b )
    break;
end
end
% clean up
if  ( i == MAX_ITERATIONS)
    warning('Maximum iterations exceeded');
end
xFinal = xN(end);
xN = xN(:);
% convert output to column vectors
```

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

Arrange \( a,b \) and \( c \) so that
- \( a \) and \( b \) form a bracket
- \( |f(b)| \leq |f(a)| \)
- \( c \) 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...

- **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.

Consider the Unit interval:

![Diagram of the Unit interval with points $u$, $v$, and $0$, $1$, $p$, $1-p$]
Golden Section Search: The Algorithm

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

\[
\begin{align*}
&\quad a \quad u \quad v \quad b \\
&\rho \quad 1-\rho
\end{align*}
\]

Initialize:

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 \( i=1:\text{MAX\_ITERATIONS} \)

\[ f = \text{func}(x); \]
\[ p = \text{polyfit}(x,f,2); \]
\[ \text{pPrime} = n.*p; \]
\[ x\text{New}(i) = -\text{pPrime}(2)/\text{pPrime}(1); \]
\[ x = [x(2:end) \ x\text{New}(i)]; \]
\[ \text{relErr} = \text{abs}(x\text{New}(i)-x\text{New}(i-1))/\text{abs}(x\text{New}(i)); \]

if \( \text{relErr} < \text{tol} \)

break

end

end

Hybrid schemes: \texttt{fminbnd}

successive Parabolic interpolation + golden section search

in \texttt{Matlab}: use \texttt{fminbnd}

syntax:

\[
\begin{align*}
\text{options} &= \text{optimset}(\text{‘disp’,\text{‘iter’})}
\text{x} &= \text{fminbnd}(@\text{func},x1,x2,\text{options})
\end{align*}
\]

examples:

\[
\begin{align*}
\text{x} &= \text{fminbnd}(\text{@}(x) \ \text{sin}(2\pi x),0,1,\text{options})
\text{x} &= \text{fminbnd}(\text{@}(x) \ -\text{humps}(x),.1,.4,\text{options})
\end{align*}
\]