Solving Nonlinear Equations

Jijian Fan

Department of Economics
University of California, Santa Cruz

Oct 13 2014

Overview

- NUMERICALLY solving nonlinear equation

$$f(x) = 0$$

- Four methods
  - Bisection
  - Function iteration
  - Newton's
  - Quasi-Newton

Motivation

Linear equation can be solved analytically

- $Ax = b \qquad \Rightarrow \qquad x = A^{-1}b$
- Methods such as L-U factorization, Gaussian elimination, etc

Motivation

Linear equation can be solved analytically

- $Ax = b \quad \Rightarrow \quad x = A^{-1}b$
- Methods such as L-U factorization, Gaussian elimination, etc

However, nonlinear equation might not be explicitly solved

- e.g. $f(x) = x^{-0.8} + 2x^{0.5} - 3 = 0$
- Numerical methods

Numerical methods

- "Continuous" means 1, 1.001, . . . , 1.999, 2
- "Equality" means $1 = 1.0003$

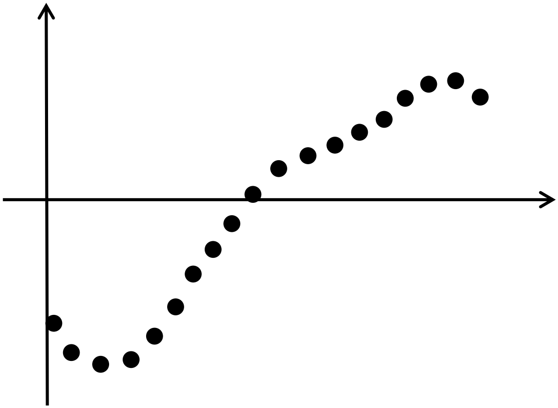

Figure 1 : A "continuous" function in computer

Bisection Method

- Based on Intermediate Value Theorem
- Start with a bounded interval $[a, b]$ such that $f(a)f(b) < 0$

- Sample code

```
while (b-a)>tol;
  if sign(f((a+b)/2)) == sign(f(a))
    a= (a+b)/2;
  else
    b= (a+b)/2;
end
x=a;
```

Bisection Method

- Advantage
  - Reliable: always finds the root
  - LEAST requirements on functional properties

- Disadvantage
  - Univariate $f : \mathbb{R} \mapsto \mathbb{R}$
  - Slow $\log(n)$

Function Iteration

- Solve for fixed point $x = g(x)$
  - $f(x) = 0 \Leftrightarrow x = g(x) = x - f(x)$
- Start with an initial guess $x^{(0)}$ s.t. $||g'(x^{(0)})|| < 1$
- Sample code

```
x=x0;
y=g(x);
while norm(y-x)>tol;
  x=y;
  y=g(x);
end
```

Function Iteration

- Advantage
  - Could be multivariate $f : \mathbb{R}^n \mapsto \mathbb{R}^n$
  - Easy-coding

- Disadvantage
  - Not reliable: require differentiability, and
  - Initial $x^{(0)}$ should be sufficiently close to a fixed point x^*
  - Only applicable to downward-crossing fixed point $||g'(x^*)|| < 1$
  - Worth trying even if one or more condition fails

Function Iteration: Extension

- Value Function Iteration (VFI)

$$V(k) = \max_{k'}\{u(c) + \beta V(k')\}$$

$$k' = f(k) - c + (1-\delta)k$$

- Rewrite as

$$V(k) = \max_{k'}\{u(f(k) + (1-\delta)k - k') + \beta V(k')\}$$

Function Iteration: Extension

- Make a grid of k
- Make an initial guess $V^0(k)$ for each k
- Updating: for every k, update

$$V^{i+1}(k) = \max_{k'}\{u(f(k) + (1-\delta)k - k') + \beta V^i(k')\}$$

  by trying each possible k'
- Repeat updating
- Until $V^{i+1}(k)$ is close enough to $V^i(k)$

Newton's Method

- First-order Taylor approximation

$$f(x) \approx f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) = 0$$

- Solve for the iteration rule

$$x^{(k+1)} \leftarrow x^{(k)} - [f'(x^{(k)})]^{-1} f(x^{(k)})$$

- Start with an initial guess $x^{(0)}$

Newton's Method

- First-order Taylor approximation

$$f(x) \approx f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) = 0$$

- Solve for the iteration rule

$$x^{(k+1)} \leftarrow x^{(k)} - [f'(x^{(k)})]^{-1} f(x^{(k)})$$

- Start with an initial guess $x^{(0)}$
- Pseudo-code

```
for iter=1:maxiter
  [ fval fjac ]=f(x);
  x = x - fjac \ fval;
  if norm(fval) < tol, break, end
end
```

Newton's Method: Calculate the Jacobian Matrix

- How to calculate the Jacobian Matrix

$$\mathrm{f}'(\mathrm{x}) = \begin{bmatrix} \partial \mathrm{f}_1/\partial \mathrm{x}_1 & \partial \mathrm{f}_1/\partial \mathrm{x}_2 & \dots & \partial \mathrm{f}_1/\partial \mathrm{x}_\mathrm{n} \\ \partial \mathrm{f}_2/\partial \mathrm{x}_1 & \partial \mathrm{f}_2/\partial \mathrm{x}_2 & \dots & \partial \mathrm{f}_2/\partial \mathrm{x}_\mathrm{n} \\ \vdots & \vdots & \ddots & \vdots \\ \partial \mathrm{f}_\mathrm{n}/\partial \mathrm{x}_1 & \partial \mathrm{f}_\mathrm{n}/\partial \mathrm{x}_2 & \dots & \partial \mathrm{f}_\mathrm{n}/\partial \mathrm{x}_\mathrm{n} \end{bmatrix}$$

- Analytical derivatives
- Numerical derivatives

Newton's Method: Calculate the Jacobian Matrix

- Analytical derivatives example: Cournot duopoly model

$$P(q) = q^{-1/\eta}$$

$$C_i(q_i) = \frac{1}{2} c_i q_i^2$$

$$\max_{q_i} \pi_i(q_1, q_2) = P(q_1 + q_2) q_i - C_i(q_i)$$

F.O.C.

$$\frac{\partial \pi_i}{\partial q_i} = P(q_1 + q_2) + P'(q_1 + q_2) q_i - C_i'(q_i) = 0$$

Let

$$\vec{f}(\vec{q}) = \begin{bmatrix} \frac{\partial \pi_1}{\partial q_1}(q_1, q_2) \\ \frac{\partial \pi_2}{\partial q_2}(q_1, q_2) \end{bmatrix}$$

Solve

$$\vec{f}(\vec{q}) = \vec{0}$$

Newton's Method: Calculate the Jacobian Matrix

Note that

$$\frac{\partial \mathrm{f_i}}{\partial \mathrm{x_j}} \equiv \frac{\partial^2 \pi_\mathrm{i}}{\partial \mathrm{q_j} \partial \mathrm{q_i}}$$

Newton's Method: Calculate the Jacobian Matrix

Note that

$$\frac{\partial \mathrm{f_i}}{\partial \mathrm{x_j}} \equiv \frac{\partial^2 \pi_\mathrm{i}}{\partial \mathrm{q_j} \partial \mathrm{q_i}}$$

```
function [fval,fjac]=f(q)
  c=[0.6,0.8]; eta=1.6; e=-1/eta;
  fval=sum(q)^e + e*sum(q)^(e-1)*q-diag(c)*q;
  fjac=e*sum(q)^(e-1)*ones(2,2)+e*sum(q)^(e-1)*eye(2)
   + (e-1)*e*sum(q)^(e-2)*q*[1 1]-diag(c);
end
```

Newton's Method: Calculate the Jacobian Matrix

- Numerical derivatives

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x)}{(x+\varepsilon) - x} = \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

Newton's Method: Calculate the Jacobian Matrix

- Numerical derivatives

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x)}{(x+\varepsilon) - x} = \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

- Centered finite difference approximation

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x-\varepsilon)}{(x+\varepsilon) - (x-\varepsilon)} = \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}$$

Newton's Method: Calculate the Jacobian Matrix

- For multivariate case, let

$$\varepsilon = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \varepsilon \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Quasi-Newton Methods

Calculating $f'(x)$ and taking inverse is

- Slow
- Inefficient

Goal

- Find a proper approximation of $f'(x)$ or $(f'(x))^{-1}$
- Update this approximation in a more efficient way

Methods

- Secant method
- Broyden's method

Secant Methods

- Univariate
- Approximate derivatives (tangent) by secant

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

thus

$$[f'(x^{(k)})]^{-1} \approx \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}$$

Secant Methods

- Univariate
- Approximate derivatives (tangent) by secant

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

thus

$$[f'(x^{(k)})]^{-1} \approx \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}$$

- Iteration rule

$$x^{(k+1)} \leftarrow x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})} f(x^{(k)})$$

- Need two initial value

Broyden's Methods

Generalized secant method for multivariate

- Denote $A^{(k)}$ as the Jacobian approximant of f at $x = x^{(k)}$
- Newton iteration

$$x^{(k+1)} \leftarrow x^{(k)} - (A^{(k)})^{-1} f(x^{(k)})$$

- Secant condition must hold at $x^{(k+1)}$

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

Broyden's Methods

- Choose $A^{(k+1)}$ that minimizes Frobenius norm

$$\min_{A^{(k+1)}} ||A^{(k+1)} - A^{(k)}|| = \sqrt{\text{trace}((A^{(k+1)} - A^{(k)})^\top (A^{(k+1)} - A^{(k)}))}$$

subject to

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

Broyden's Methods

- Choose $A^{(k+1)}$ that minimizes Frobenius norm

$$\min_{A^{(k+1)}} ||A^{(k+1)} - A^{(k)}|| = \sqrt{\mathrm{trace}((A^{(k+1)} - A^{(k)})^\top (A^{(k+1)} - A^{(k)}))}$$

subject to

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

- Solve for $A^{(k+1)}$

$$A^{(k+1)} \leftarrow A^{(k)} + [f(x^{(k+1)}) - f(x^{(k)}) - A^{(k)} d^{(k)}] \frac{{d^{(k)}}^\top}{{d^{(k)}}^\top d^{(k)}}$$

where

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

Broyden's Methods

- Improvement: directly update $\mathrm{B}^{(\mathrm{k})} \equiv (\mathrm{A}^{(\mathrm{k})})^{-1}$
- Sherman-Morrison formula

$$(\mathrm{A} + \mathrm{u}\mathrm{v}^\top)^{-1} = \mathrm{A}^{-1} + \frac{1}{1 + \mathrm{u}^\top \mathrm{A}^{-1}\mathrm{v}} \mathrm{A}^{-1}\mathrm{u}\mathrm{v}^\top \mathrm{A}^{-1}$$

Broyden's Methods

- Improvement: directly update $B^{(k)} \equiv (A^{(k)})^{-1}$
- Sherman-Morrison formula

$$(A + uv^\top)^{-1} = A^{-1} + \frac{1}{1 + u^\top A^{-1} v} A^{-1} u v^\top A^{-1}$$

- Iteration rule

$$B^{(k+1)} \leftarrow B^{(k)} + \frac{(d^{(k)} - u^{(k)}) {d^{(k)}}^\top B^{(k)}}{{d^{(k)}}^\top u^{(k)}}$$

where

$$d^{(k)} = x^{(k+1)} - x^{(k)} \qquad u^{(k)} = B^{(k)}[f(x^{(k+1)}) - f(x^{(k)})]$$

Broyden's Methods

- Pseudo-code

```
Choose initial x
Calculate initial B (usually B = f'^{-1}(x))
loop
  update x
  if f(x) is close enough to 0 then break
  update B
end
```

Summary

- Four method to solve nonlinear equations
  - Bisection: robust but relatively slow
  - Function iteration: easy-coding
  - Newton & Quasi-Newton: quick, most popular but not always work
- May not work for the multi-root case