NAKE Workshop
June 11, 2001
Numerical Methods for Dynamic Economic Analysis
Ken Judd

Introduction

Observations:

- Closed-form solutions to economic models are rare.
- Qualitative analyses fail to address key issues.

Objective:
Develop numerical techniques which are

- Generally useful for
  1. macroeconomics
  2. public finance
  3. industrial organization
  4. finance
  5. game-theoretic analyses
- Accessible, using
  1. general techniques from numerical analysis
  2. standard software
  3. widely available hardware
- Fast, usable
  1. in estimation
  2. for computing comparative statics and dynamics

Some Examples:

- Computational General Equilibrium (Scarf, Shoven, Whalley, Johansen, Rutherford, Ferris)
- Agricultural Economics: Gustafson(1958), Williams and Wright (EJ 1982, QJE 1984, 1990), Miranda and Helmburger (AER 1988)
- Games: Howson-Lemke, Wilson
- Macroeconomics: Taylor and Uhlig (JBES 1990 symposium)

Current state-of-the-art:

- Best algorithms are for static models
- Need solution methods for dynamic models
- Current methods for dynamic, stochastic, competitive general equilibrium models are slow
- Few methods available for dynamic, closed-loop (a.k.a., Markov perfect) Nash equilibria
- Very little available for supergame

Strategy: Find out what physicists, chemists, and engineers do

- Economic problems often are parabolic PDE's:

  $$y_t = f(y_{xx}, y_x, y, x, t)$$

  similar to many physics problems

- Two general methods in science:

  1. Perturbation Methods - like Taylor series expansions
  2. Projection Methods – like doing econometrics

Outline of these lectures

Monday:

Basic numerical methods: errors, linear equations, optimization, nonlinear equations

Tuesday:

Approximation, integration, dynamic programming

Wednesday:

Projection method

Thursday:

Perturbation methods

Friday:

Bifurcation methods

Summary: the role of computation in economics

Numerical Solution vs. Theoretical Solution

Source of Errors

- finite-state nature of machines
  - On a five-digit machine

$$\begin{aligned} 1.000245 - 1.000100 &\doteq 1.0002 - 1.0001 \\ &= .0001 \\ &\neq .000145 \end{aligned}$$

  - double precision means that we have 16-digit accuracy – NOT ENOUGH!
- finite-time constraint on computation
  - Cannot compute infinite series, so we compute finite truncations:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \doteq \sum_{n=0}^{N} \frac{x^n}{n!}$$

  - Many algorithms compute infinite sequences x_k which converge to truth, but we must stop at some finite k.

Approach to real problems

- establish acceptance criterion, similar to ϵ-equilibrium
- find some candidate which satisfies acceptance criterion

Linear Equations

The most basic task in numerical analysis is solving the linear system of equations

$$Ax = b$$

where $b \in R^n$ and $A \in R^{n \times n}$.

- Importance
  - Linear equations arise frequently
  - Nonlinear problems are often reduced to a sequence of linear problems
- Types of problems
  - A is dense if $a_{ij} \neq 0$ for most i, j.
  - A is sparse if $a_{ij} = 0$ for most i, j.
- Methods
  - LU decomposition – classic method
  - QR decomposition
  - Cholesky decomposition – for symmetric A
- Software
  - Simple commands in Matlab, Gauss
  - Lapack – Fortran and C

Error Analysis

- Matrix norm: If $\| \cdot \|$ is a norm on R^n, then the induced norm of A is

$$\| A \| \equiv \max_{x \neq 0} \frac{\| Ax \|}{\| x \|} = \max_{\|x\|=1} \| Ax \|$$

- Spectral radius:
  - $\sigma(A) = \{\lambda | \lambda \text{ is an eigenvalue of } A\}$
  - $\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|$
- Theorem: For any norm $\| \cdot \|$, $\rho(A) \leq \| A \|$.
- Sensitivity of the solution of $Ax = b$ to errors
  - Source of error
    * roundoff error in b, A.
    * b comes from a numerical procedure which has error
    * errors can build up to be nontrivial
  - Error model: $A(x + e) = (b + r)$
  - Elasticity of error:

$$\mathcal{E} = \frac{\| e \|}{\| x \|} \div \frac{\| r \|}{\| b \|}$$

- Condition number:

$$\text{cond}(A) \equiv \ \parallel A \parallel \ \ \parallel A^{-1} \parallel$$

  - If $A = aI$, then $Cond(A) = \mathcal{E} = 1$.
  - $Cond(A)$ is a rough approximation of $\mathcal{E}$.
  - $Cond(A)$ is useful measure of being singular
    - If $A = \epsilon I_n$, then $\text{cond}(A) = 1$, but $\det(A) = \epsilon^n$
    - If $\det(A) = 0$, then $\text{cond}(A) = \infty$
  - Spectral condition number is used since

$$\text{cond}(A) \geq \frac{\max_{\lambda \in \sigma(A)} |\lambda|}{\min_{\lambda \in \sigma(A)} |\lambda|} \ \equiv \ \text{cond}_* (A),$$

    where $Cond_*(A)$ is the spectral condition number of A.

- Error elasticities and conditions are often large
  - $Cond(A)$, and $\mathcal{E}$ are usually at least 10^5 – construct random A and average $\text{cond}_* (A)$ is easily 10^5.
  - $\text{Log}_{10} Cond(A)$ indicates loss of precision - number of lost digits – $\mathcal{E} = 10^5$ means x has 5 fewer correct digits than b.
  - $Cond(A)$, and $\mathcal{E}$ can easily be 10^{10} – x has 10 fewer correct digits than b!

- Lesson: Errors can easily arise and grow when solving a sequence of linear equations.

Optimization

Many economic problems are optimization problems:

$$\begin{array}{rl} \min_x & f(x) \\ s.t. & g(x) = 0, \\ & h(x) \leq 0, \end{array}$$

where

- $f : R^n \to R$ is the objective function,
- $g : R^n \to R^m$ is the vector of m equality constraints
- $h : R^n \to R^\ell$ is the vector of ℓ inequality constraints.

Newton's Method: Unconstrained Problems

- Define gradient and Hessian of f:

$$\begin{aligned} \bigtriangledown f(x) &= \left(\frac{\partial f}{\partial x_1}(x), \cdots, \frac{\partial f}{\partial x_n}(x)\right) \\ H(x) &= \left(\frac{\partial^2 f}{\partial x_i \, \partial x_j}(x)\right)_{i,j=1}^{n} \end{aligned}$$

  – Replace $f(x)$with a quadratic approximation of f based at x^k:

$$f(x) \doteq f(x^k) + \bigtriangledown f(x^k)(x - x^k) + \frac{1}{2}(x - x^k)^\top H(x^k)(x - x^k)$$

  – The next iterate in Newton's method is minimum of quadratic approximation:

$$x^{k+1} = x^k - H(x^k)^{-1}(\bigtriangledown f(x^k))^\top .$$

Theorem 1 *Suppose that $f(x)$ is minimized at x^*, C^3 in a neighborhood of x^*, and that $H(x^*)$ is nonsingular. If x_0 is close enough to x^*, then Newton's method converges quadratically to x^*; that is,*

$$\lim_{n\to\infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^2} < \infty$$

- Newton plus linesearch chooses best point in Newton direction:

$$\begin{aligned} s^k &= -H(x^k)^{-1}(\bigtriangledown f(x^k))^\top \\ \lambda &= \arg\min f(x^k + \lambda s^k) \\ x^{k+1} &= x^k + \lambda \, s^k \end{aligned}$$

Stopping Rules

Stop iteration when

- Sequence is not changing much

$$\| x^{k+1} - x^k \| < \epsilon(1 + \| x^k \|)$$

- Gradient is close to zero

$$\| \bigtriangledown f(x^k) \| \leq \delta(1 + |f(x^k)|).$$

Performance of Newton's method

- Newton method tradeoffs:
  - Quadratic convergence when close to solution – few iterations often suffice
  - Linesearch version will always converge to a local minimum
  - Hessian is expensive to compute: $O(n^3)$
  - Hessian may be expensive to store: $O(n^2)$
- Solutions to problems
  - Compute approximations to Hessian
  - If Hessian is to big, then just search in a useful sequence of directions - conjugate gradient method

Nonlinear Least Squares

Canonical problem:

$$\min_x \frac{1}{2} \sum_{k=1}^{m} f^k(x)^2 \equiv S(x),$$

- Could use standard optimization procedure
- Special procedure: Gauss-Newton
  - Define $J(x)$ to be the Jacobian matrix of $f(x) \equiv (f^1(x), \ldots, f^m(x))^\top$:
    $$J^i_\ell(x) = \left(\frac{\partial f^i}{\partial x_\ell}\right)$$
  - Gradient of $S(x)$ is $J(x)^\top f$,
  - Hessian of $S(x)$ is $J(x)^\top J(x) + G(x)$, where $G(x)$ is often small
  - The $J(x)^\top J(x)$ piece of the Hessian is easy to compute
  - Gauss-Newton algorithm: take $J(x)^\top J(x)$ as estimate of Hessian
    $$x^{k+1} = x^k - (J(x^k)^\top J(x^k))^{-1}(\nabla f(x^k))^\top.$$
  - $J(x)^\top J(x)$ is likely to be poorly conditioned. Solution: Levenberg-Marquardt algorithm uses $J(x)^\top J(x) + \lambda I$ as an estimate of the Hessian for some scalar λ

Constrained Nonlinear Optimization

- Most economic optimization problems involve either linear or nonlinear constraints

$$\begin{array}{ll} \min_x & f(x) \\ s.t. & g(x) = 0 \\ & h(x) \leq 0 \end{array}$$

- Penalty Function Approach
  - Construct the penalty function

$$\mathcal{P}(x) = f(x) + \frac{1}{2}P\left(\sum_i \left(g^i(x)\right)^2 + \sum_j \left(\max[0, h^j(x)]\right)^2\right)$$

    where $P > 0$ is the penalty parameter.
  - If P is large, the unconstrained minimum of $\mathcal{P}(x)$ is close to solution of constrained problem
  - Basic idea is difficult to implement efficiently - use good software

Global Optimization

- Many problems have several solutions
- Many problems have local minima (maxima) which are not global solutions
- Solution: restart conventional algorithm at several initial guesses

Software

Use professionally written software; do NOT write your own

- Free packages
  - MINPACK
  - CONOPT
  - SLAC library
- Commercial
  - NPSOL - excellent
  - MINOS - great for problems with sparse constraints
  - SNOPT - excellent
  - GAMS - user friendly, contains MINOS
  - NAG - contains NPSOL, MINPACK
  - IMSL - has routines like NPSOL, MINPACK
  - Matlab optimization toolbox - mediocre
  - Gauss

Nonlinear Equations

- Nonlinear equations in economics
  - Partial equilibrium: $D(p) = S(p)$
  - General equilibrium: $E(p) = 0$
  - Nash equilibrium: $x = R(x)$
- Nonlinear equations arise in the numerical methods used to solve dynamic models.

Simple Methods from Economic Intuition

- "Hog cycle" iteration: partial equilibrium
  - Guess p^0
  - Compute $q^1 = S(p^0)$; compute $p^1 = D^{-1}(q^1)$
  - Compute $q^2 = S(p^1)$; compute $p^2 = D^{-1}(q^2)$
  - etc.
  - p^k may converge to equilibrium ("stable") or may not
  - Also known as fixed-point iteration since $p^{k+1} = D^{-1}(S(p^k))$
- Best reply iteration: Nash equilibrium in a Cournot model:
  - Equilibrium

$$\begin{aligned} q_1 &= R_1(q_2) \\ q_2 &= R_2(q_1) \end{aligned}$$

  - Best reply iteration
    * Guess q_1^0
    * Compute

$$\begin{aligned} q_2^1 &= R_2(q_1^0) \\ q_1^1 &= R_1(q_2^1) \\ q_2^2 &= R_2(q_1^1) \\ q_1^2 &= R_1(q_2^2) \\ &\vdots \end{aligned}$$

  - (q_1^k, q_2^k) may converge to Nash equilibrium ("stable equilibrium") or may not.
  - Similar to a Gauss-Seidel method

- Tatonnment: General equilibrium
  - Adjust prices in proportion, λ, to excess demand
    $$p^{k+1} = p^k + \lambda E(p^k)$$
  - Unreliable since we know tatonnement is unreliable
- Methods from economic intuition are
  - Unreliable
  - Slow even when they do work

Newton's Method for Multivariate Equations

- Sequential solution of linear problems
  - Make initial guess x^0
  - Compute linear approximation

    $$f(x) \doteq f(x^0) + J(x^0)\,(x - x^0)$$

  - Solve for the zero of linear approximation:

    $$x^1 = x^0 - J(x^0)^{-1}\, f(x^0)$$

  - Iterate:

    $$x^{k+1} = x^k - J(x^k)^{-1}\, f(x^k)$$

- Quadratic convergence if initial guess is good
- May not converge
  - Can usually fix this by Powell hybrid (a.k.a. Armijo method)
  - Can always fix this by using homotopy method
- Jacobian is costly to compute - fix this by approximations of Jacobian

Summary

- Numerical errors are important
  - Unavoidable in real computers
  - Simple arithmetic operations cause errors
  - Errors will accumulate as an algorithm proceeds
- Must use methods which minimize the buildup of errors
  - Allow iterative schemes to proceed until truncation error is small
  - Avoid ill-conditioned matrices when solving linear problems
- Economic intuition is not a source for reliable algorithms
  - Simple algorithms converge slowly at best
  - Economics is good for economic problems, not for numerical problems
- Numerical analysis methods are available to solve problems.