

IAP Lecture: Numerical Methods in Economics

Stephen P. Ryan

MIT

January 26, 2006

Overview of Talk

- Platforms for econometric and simulation work: Programming languages versus statistical packages
- Easy numerical techniques to make your life easier: numerical integration (Monte Carlo and quadrature), differentiation, and dynamic programming
- Solution techniques for optimization: Laplace-type estimators

Platforms for Numerical Work

- Factors important in platform choice:
 - Execution speed
 - Ease of use
 - Portability
 - Power
 - Learning curve
 - Long-run viability
- Two approaches: Canned packages (runtime interpreted code) and programming languages (compiled code)

Canned Statistical Packages

- Examples: Stata, S-Plus, Matlab
- Benefits:
 - At least partial GUI interface, easy to use
 - Low learning curve
 - Positive network effects across researchers (co-authors, large library of routines)
 - Very powerful data manipulation, graphing
- Costs:
 - Very slow unless you perform everything as vectorized operations (minimal loops)
 - Programming large programs difficult
 - Packages are expensive

Programming Languages

- Examples: Java, C, Fortran
- Benefits:
 - Fast
 - Flexible
 - Powerful
- Costs:
 - Learning curve
 - Larger fixed costs to even small problems
 - Harder to collaborate with colleagues
 - Less likely someone has coded up an econometric technique already

My Platforms

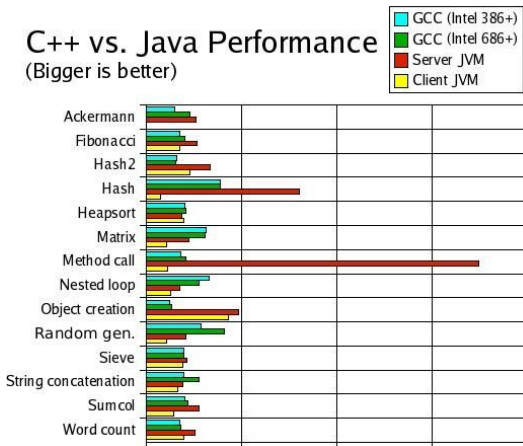
- Stata for data manipulation, summary statistics, simple regressions
- Java for everything else
 - java.sun.com for Java foundation (compiler, libraries, executor)
 - netbeans.org for integrated development environment (IDE)
 - JAMA (math.nist.gov/javanumerics/) for matrix library
 - Ports of Fortran optimization libraries (UNCMIN, MINPACK) at FPL Statistics Group (<http://www1.fpl.fs.fed.us/optimization.html>)

Why Java?

- Cost: Free!
- Excellent documentation
 - Documentation system part of language
 - Support for Javadoc built straight into IDE
 - Java Tutorial at java.sun.com goes from basics to most advanced
- Mature, highly supported
 - Many books, forums, online code examples
 - Good, free libraries for matrix algebra, gradient-based optimization
- Modern
 - Object-oriented
 - Automatic memory allocation
 - Dynamic optimization of execution paths
 - Designed from day one to be networked, multithreaded, write-once-run-anywhere cross-platform

How Fast Is It?

- Answer depends, but sometimes competitive with C++:



- Key point: 10-1000 times faster than Matlab!

Some Numerical Tools

- Numerical integration
 - Solving integrals is common in econometrics (computing expected values)
 - Monte Carlo methods and quadrature methods
- Numerical Differentiation
- Dynamic programming

Numerical Integration – Monte Carlo

- Object: compute $E[f(x)] = \int_0^1 f(x)dx$, where $x \sim U[0, 1]$.
- Crude Monte Carlo: $E[f(x)] = \frac{1}{N} \sum_{i=1}^N f(x_i)$, where x_i is drawn from $U[0, 1]$
- High variance, requires a large number of draws to get precise estimates
- Stratified sampling: variance over subinterval lower than over whole; divide $[0, 1]$ into $[0, \alpha]$ and $[\alpha, 1]$:

$$E[f(x)] = \frac{\alpha}{N} \sum_i f(x_{1i}) + \frac{1-\alpha}{N} \sum_i f(x_{2i})$$

- Antithetic variates: If f is monotone, $f(x)$ and $f(1-x)$ negatively correlated: $E[f(x)] = \frac{1}{2N} \sum_{i=1}^N (f(x_i) + f(1-x_i))$
- Also importance sampling, $E[f(x)] = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}$.

Numerical Integration – Quadrature

- Quadrature takes a different approach to the same problem
- Newton-Cotes formula: $\int_a^b f(x)dx = \sum_{i=1}^N \omega_i f(x_i)$
- Simplest quadrature is the midpoint formula, along with trapezoid rule, Simpson's rule (rules define ω_i , arbitrary x_i)
- Gaussian quadrature makes efficient choices for both ω_i and x_i
- Gaussian quadrature integrates polynomials of degree $2n - 1$ exactly using n nodes and n weights
- General enough to be applied to problem of computing $\int_a^b f(x)w(x)dx$
- Some special problems have weights and nodes worked out for us already

Numerical Integration – Quadrature

- Simplest example, $\omega(x) = 1$, gives rise to Gauss-Legendre quadrature formula:

$$\int_{-1}^1 f(x) dx = \sum_{i=1}^n \omega_i f(x_i) + \frac{2^{2n+1} (n!)^4}{(2n+1)! (2n)!} \cdot \frac{f^{(2n)}(\xi)}{(2n)!}$$

- After applying Sterling's formula, we can show that the error term is bounded above by

$$\pi 4^{-n} \left(\sup_m \left[\max_{-1 \leq x \leq 1} \frac{f^{(m)}(x)}{m!} \right] \right)$$

- The term in brackets is finite for most functions, meaning rate of convergence is exponential in the number of quadrature nodes

Numerical Integration – Quadrature Example

- Example from Judd (2001)
- Error in computing $-\int_0^{50} e^{-0.05t} \left(1 + \frac{t}{5} - 7\left(\frac{t}{50}\right)^2\right)^{1-\gamma} dt$ using Gauss-Legendre quadrature:

Number of nodes	$\gamma = 0.5$	$\gamma = 1.1$
3	5(-3)	2(-3)
5	1(-4)	8(-5)
10	1(-7)	1(-7)
15	1(-10)	2(-10)
20	7(-13)	9(-13)

Other common economic problems

- Other problems have special forms
- If $\omega(t) = \exp(-x)$, as in exponential discounting, then we can use *Gauss-Laguerre* quadrature:

$$\int_a^\infty e^{-ry} f(y) dy \approx \frac{e^{-ra}}{r} \sum_{i=1}^n \omega_i f\left(\frac{x_i}{r} + a\right)$$

- If $\omega(t) = e^{-x^2}$, as when integrating out a normal variable, then we can use *Gauss-Hermite* quadrature:

$$E[f(Y)] \approx \pi^{-1/2} \sum_{i=1}^n \omega_i f(\sqrt{2}\sigma x_i + \mu),$$

where Y is a normal variable with mean μ and variance σ^2

- Nodes have been pre-computed, e.g. see Judd (2001)

Numerical Differentiation

- Exact differentiation is always preferred to numerical differentiation, due to approximation errors
- Sometimes that isn't practical, have to use finite differences
- Forward difference formula:

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

- However, the best numerical accuracy is $\sqrt{\epsilon_f}$, where ϵ_f is machine accuracy
- If we use symmetric differences, we can improve accuracy by two orders of magnitude (100 times!) better

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

- If going to be evaluating same function repeatedly, can approximate it via Chebyshev approximation

Dynamic Programming

- Objective: solve

$$V_i = \max_u \left[\pi(x_i, u) + \beta \sum_{j=1}^n q_{ij}(u) V_j \right], i = 1, \dots, n$$

where there are n states, and q_{ij} denotes the probability of transition between state i and j , conditional on policy u .

Theorem (Denardo, 1967)

If X is compact, $\beta < 1$, and π is bounded above and below, then the map $TV = \sup_{u \in D(x)} \pi(x, u) + \beta E(V(x^+) | x, u)$ is a contraction mapping with modulus β and a unique fixed point.

- This means we can find the unique solution to the dynamic program using the iterated mapping

Value Function Iteration

- Simplest method is *Value Function Iteration*:
 - 1 Make initial guess V^0 .
 - 2 For $i = 1, \dots, n$, compute $V_i^{l+1} = \max_{u \in D} \pi(x_i, u) + \beta \sum_{j=1}^n q_{ij}(u) V_j^l$.
 - 3 If $\|V^{l+1} - V^l\| < \epsilon$, stop. Otherwise repeat step 2.
- Structure allows us to guarantee that solution is within ϵ^V of the true solution if

$$\|V^{k+1} - V^k\| \leq \epsilon^V (1 - \beta)$$

- Typically just set ϵ^V to be a really small number
- Easy to implement, but can requires lots of iterations; want to accelerate convergence process

Speeding Up Convergence

- One alternative is *Policy Function Iteration*:
 - 1 Make initial guess V^0 .
 - 2 Solve for optimal policies, given V^l : $U^{l+1} = \mathcal{U}V^l$
 - 3 Compute vector of per-period payoffs, given policies:
 $P_i^{l+1} = \pi(x_i, U_i^{l+1}), i = 1, \dots, n.$
 - 4 Compute the new value function through matrix algebra:
 $V^{l+1} = (I - \beta Q^{U^{l+1}})^{-1} P^{l+1}.$
 - 5 If $\|V^{l+1} - V^l\| < \epsilon$, stop. Otherwise repeat step 2.
- May only take a few iterations, but inversion may be computationally very expensive and slow

Speeding Up Convergence

- Another approach is to borrow from solving systems of equations
- *Gauss-Jacobi* involves a slightly modified update of the value function:

$$V_i^{l+1} = \max_u \frac{\pi(x_i, u) + \beta \sum_{j \neq i} q_{ij}(u) V_j^l}{1 - \beta q_{ii}(u)}$$

- By same intuition, *Gauss-Seidel* can start using new information in updates as soon as it is available:

$$V_i^{l+1} = \max_u \frac{\pi(x_i, u) + \beta \sum_{j < i} q_{ij}(u) V_j^{l+1} + \beta \sum_{j > i} q_{ij}(u) V_j^l}{1 - \beta q_{ii}(u)}$$

Optimization Problems

- Perhaps the most common numerical problem is the minimization of a function
- Goal: find $\min_x f(x)$
- This covers just about everything, since you can cast most problems into a minimization problem (GMM, maximum likelihood, computing Nash equilibria, solving systems of equations)
- This problem is really difficult in general, since there may be lots of local equilibria, nonexistence of derivatives, and $f(x)$ may be really expensive to compute
- Several solutions have been advanced to deal with these problems; we'll look at Laplace-type estimators (LTE's)

Laplace-type Estimators

- Due to Chernozhukov and Hong (2003)
- How it works: Apply MCMC sampling methods to find minima of hard-to-minimize functions, and trace out quasi-posterior distributions of the parameters at the same time
- Key point: The LTE method generates new guesses for the parameter vector from a *proposal density*, and accepts or rejects that new point based on the improvement in the objective function
- Starting with a guess for the parameter vector and a proposal density, algorithm runs in three steps
 - 1 Conduct a long “burn-in” to find a (local) minimum
 - 2 Tune the proposal density so that the algorithm generates new values in the “right” neighborhood
 - 3 Generate the long Markov chain used for inference

Laplace-type Estimators

- Benefits:
 - Easy to implement
 - Robust to local minima, flat spots, jumps in objective function
 - Computes *all* posterior moments of parameters simultaneously (e.g. percentiles, median, mean)
 - Applicable to wide range of statistical criterion functions (GMM, empirical likelihood, etc.)
 - Also useful as a general minimization tool
- Drawbacks:
 - Requires large numbers of function evaluations ($\geq 20,000$)
 - Tuning process can bog down, get lost for high variance parameters
- Other approaches: simulated annealing and differential evolution

Conclusion

- Ken Judd's book "Numerical Methods in Economics" is a divine resource for applied researchers
- Choice of right tool for the job can really save you a lot of headaches
- A little work to get some advanced computational tools setup reaps recurring benefits as they can be used repeatedly
- Better numerical tools open up more realistic modeling opportunities, expand the reach of questions that we can address