Finding all solutions to discrete games (using Stata)

Professor Matthew J. Baker

February 12, 2018

Overview

- I present and describe some tools and ideas for solving discrete action games.
- Along the way, a bit about games and also solving equation systems.
- My hope is that you will do this and help me trouble shoot the routines before I go fully public!
- You also may learn a little bit about numerical solution methods.

Questions

Why would someone want to have tools to solve games?

1. Practical interest: finding (all) Nash equilibria - and having guarantees that one has done so - is a difficult computational problem (in NP class or worse?). What if one wishes to solve a sequence of games?
2. Why in Stata? Why not use *Gambit* - a freely available solver? *Gambit* is hard to integrate with estimation, and sometimes crashes.
3. Interest in integrating game theory with econometrics (for example, Bajari, Hong, and Ryan (2010, *Econometrica*).
4. The idea: posit that a sequence of observed market outcomes are the result of a game played by competitors. Walmart, Kmart, Target entry example.

Moreover...

It is also of some independent interest - solving a sequence of games also can help students and researchers get a feel for how the number and nature of equilibria change as:

1. The number of players change
2. Parameters change

One can also think about the expected number of solutions to games of particular types. (i.e., coordination games with some sort of randomized returns)

Why is the problem difficult?

Like many things, %95 of the problem is (somewhat) simple. The rest is (really) hard and this occupies a lot of time and effort! Difficulties:

1. Mixed strategies: Generally involve finding all solutions to polynomial equations!
2. "Support enumeration" for the purposes of finding partially (all) mixed strategies.

These two together mean the computational task rapidly grows as the number of players and number of actions expands. NP hard or worse?

Discrete action games

- **Discrete action game:** a game with an arbitrary number of players $N \geq 2$ each with A_i potential actions, where $\#A_i \in 1, 2, 3, ...$
- **Solutions:** Nash equilibrium strategy profiles for the complete information, simultaneous move game.
- This includes pure, totally mixed, and partially mixed equilibrium strategy profiles.
- The mixed-strategies present the difficulties! While one can ignore them, Nash's classic proof on existence of equilibrium depends upon allowing for them, as does **Wilson's "oddness" theorem**!

Details and definitions

- Let player's actions be either $s_i = 0$ or $s_i = 1$. A strategy for player i: a probability $\sigma_i \in [0, 1]$ that $s_i = 0$ is played.
- A *Nash equilibrium* is a strategy profile $\sigma_i^*, i = 1, 2, 3, \ldots, N$ where: $\pi_i(\sigma_i^*, \sigma_{-i}^*) \geq \pi_i(\sigma_i', \sigma_{-i}^*)\ \forall \sigma_i'$.
- No player i can improve payoffs by a unilateral deviation from the equilibrium strategy profile.
- Note that this definition allows mixed strategies - common knowledge probabilistic play.

Mixed strategies and existence

Mixed strategies are sometimes hard to interpret, but they are necessary for proof of Nash's celebrated theorem, which relies on fixed point arguments. A simplified version:
Define

$$g_i = \max[\pi_i(1, \sigma_{-i}) - \pi_i(0, \sigma_{-i}), 0]$$

$$\hat{\sigma}_i = \frac{\sigma_i + g_i(\sigma)}{1 + g_i(\sigma)}$$

A continuous function from $[0,1]^N$ into itself. By the Brouwer FPT, it has a fixed point in $\hat{\sigma}_i = \sigma_i$ in $[0,1]^N$. $\exists$ at least one equilibrium.

More on existence

The existence theorem - and Wilson's related "oddness" theorem – rely on mixed strategies! This makes things hard because:

- Mixed strategies involve solutions to polynomial systems.
- This is not an easy problem.
- The difficulty increases rapidly in the number of players.

Example - Entry Game

An example of a game in normal(strategic) form:

- P1 (player 1) chooses rows, while P2 (player 2) chooses columns. Strategies are either `Enter` or `StayOut`
- Payoffs are listed so that P1's payoffs are first, followed by P2's payoffs.
- Normal form: a listing of payoffs and action combinations.

P1↓,P2→	`Enter`	`StayOut`
`Enter`	$\pi - \delta, \pi - \delta$	$\pi, 0$
`StayOut`	$0, \pi$	0,0

Example - Illustration of solution techniques

P1↓,P2→	0	1
0	1,1	$0, a$
1	$b, 0$	c, c

- Dominant strategy: if $b > 1, c > 0$, 1 is a dominant strategy. 1 never plays 0
- If $b = a = 0$, $c > 0$, 00 and 11 are pure strategy equilibria.
- Mixing in previous case: $\sigma_1(1) = (1 - \sigma_1)c$. If both players do this, they are indifferent between actions. An equilibrium.

Example commentary

- In the entry (econometrically important) example, we may have one or three equilibria depending upon π and δ. If $\delta > \pi$, there are three equilibria.
- Simple, right? But what if we have three players, where payoffs for entering are: $\pi - \delta(N_e - 1)$?
- What about 5 players, where payoffs for each are: $\pi_i - \delta_i(N_e - 1)$
- The last permutation hints at how the econometrics might work; we might specify that $\pi_i = X\beta_i$ and try to estimate β, δ based on what we observe.

A second example

Games are usually described in terms of the *normal* or *extensive* form. However, I have found it computationally better to use a "list" form.

- Enumerate pure-strategy profiles exhaustively using ones and zeros.
- Enumerate corresponding payoffs.
- This format tends to mesh nicely with computer programming and parallel computation.

Second example continued

Let's call the action profiles a matrix **A** and the payoffs a matrix **Π**. Let `Enter` be denoted as action 0 and `StayOut` be denoted as action 1. In the three-player case, we have:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The matrix **Π** would have the same dimensions (always $N \times 2^N$) and includes corresponding payoffs.

A full example

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{\Pi} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1.2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

This is a three-player coordination game. It has three equilibria.

Solving games

There are really three ways to proceed (following the second example):

1. Pare down the game by elimination of dominated strategies.
2. Test all possible action profiles for pure-strategy equilibria (can be time-consuming).
3. Solve polynomial systems for totally mixed and partially mixed equilibria - or verify that they are not there. Note that this is much harder for games where $N > 2$ than it is for $N = 2$.

An illustration - polynomial system for 3-player case

Let the probabilities be σ_1,σ_2, and σ_3. Then we have to solve:

$$\sigma_2\sigma_3 a_{000} + (1-\sigma_2)\sigma_3 a_{010} + \sigma_2(1-\sigma_3)a_{001} + (1-\sigma_2)(1-\sigma_3)a_{011} = \sigma_2\sigma_3 a_{100} + (1-\sigma_2)\sigma_3 a_{110} + \sigma_2(1-\sigma_3)a_{100} + (1-\sigma_2)(1-\sigma_3)a_{111},$$
$$\sigma_1\sigma_3 b_{000} + (1-\sigma_1)\sigma_3 b_{100} + \sigma_1(1-\sigma_3)b_{001} + (1-\sigma_1)(1-\sigma_3)b_{101} = \sigma_1\sigma_3 b_{010} + (1-\sigma_1)\sigma_3 b_{110} + \sigma_1(1-\sigma_3)b_{011} + (1-\sigma_1)(1-\sigma_3)b_{111},$$
$$\sigma_1\sigma_2 c_{000} + (1-\sigma_1)\sigma_2 c_{100} + \sigma_1(1-\sigma_2)c_{010} + (1-\sigma_1)(1-\sigma_2)c_{110} = \sigma_1\sigma_2 c_{001} + (1-\sigma_1)\sigma_2 c_{101} + \sigma_1(1-\sigma_2)c_{011} + (1-\sigma_2)(1-\sigma_2)c_{111}$$

Or verify that there is no solution. This is not counting partial mixing.

How is a polynomial system to be solved?

How should one go about solving a system above, or, more correctly:

1. Find all possible solutions, and
2. be sure that one has them all?

Really, three possibilities:

- **Homotopy methods** - theoretically elegant, but numerically unstable. Involves complex numbers.
- **Algebraic methods** (Groebner bases, etc) - extremely theoretically elegant, extremely numerically unstable, involve complex numbers.
- **Interval methods** - brute force, numerically stable, but sometimes slow. Only real solutions found.

Anyone want to guess which method I use?

A seven-slide description

Basic idea of interval methods: one has a function $f(x)$ and a variable x ranging over some interval $[a, b]$. We want to know: where in $[a, b]$ do we have $f = 0$? Is there such a point? Interval analysis says:

- given x ranges over $[a, b]$, what are the range of values that f could take on?
- And is 0 included in this range?
- So, one uses "outward rounding" on all computations to be sure about any conclusions.

Basic operations

Really, only three operations and a theorem needed to get where we need to go:

- Addition: $[a, b] + [c, d] = [a + c, b + d]$
- Subtraction: $[a, b] - [c, d] = [a - d, b - c]$
- Multiplication:
  $[a, b] * [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$
- Note how computation involves operations on the upper and lower bounds only!

If we have a function composed of addition, subtraction, and multiplication, we can replace $+, -, *$ with their interval counterparts and get a so-called *natural inclusion function*. This is usually called $[f]([x])$.

Inclusion functions

Some facts about inclusion functions:

- They are convergent - as the size of the interval shrinks, so does the range $[f]$.
- But they carry a property called *pessimism*. While it is true $f([x]) \in [f]([x])$, it is usually not the case that $f([x]) = [f]([x])$, and in fact it matters how the function is written.
- Example: consider $f(x) = x^2$, $x \in [-1, 2]$. Evidently $f([x]) = [1, 4]$, as is $[f]([x])$. However, if we write $f(x) = xx$, then $[f]([x]) = [-2, 4]$, which contains the "right" answer but also is "pessimistic."

Generally, the more arguments or appearance of x, the greater the pessimism.

Finding zeros with inclusion functions

Suppose we want to find out if there is an x' in $[x] = [a, b]$ where some function is zero, and we want to find this point. Using intervals, we start with a list of boxes $[x]$ that completely cover the possible range:

1. test whether $0 \in [f]([x])$. If not, discard the box.
2. If so:
   1. Contract $[x]$ as much as possible to get a smaller box $[x']$. If $[x']$ is small enough, add it to the list of solutions.
   2. If $[x]$ is too large, bisect $[x]$ into smaller intervals and add the new boxes to the list.
   3. Go back to step 1.
3. Keep doing this until there are no intervals left to be considered. Then, apply newton iteration to find a point.

An example

Suppose we want to find all the zeros of the function $f(x) = x^2 - 6x + 8$ over the range $[0, 5]$.

1. Step 1. We find that for $[0, 5]$, $[x][x] - 6[x] + 8 = [-22, 33]$
2. Step 2. Split interval into $[0, 2.5], [2.5, 5]$. The function now has $[-7, 14.25]$, $[-15.75, 18]$.
3. Step 3. Four intervals: $[0, 1.25]$,$[1.25, 2.5]$,$[2.5, 3.75]$,$[3.75, 5]$. Now the values are: $[.5, 9.5625]$,$[-5.4375, 6.75]$, $[-8.25, 7.0625]$, $[-7.935, 10.5]$. *We can drop the box* $[0, 1.25]$! Bisect the others and repeat.
4. Repeat until all remaining boxes are smaller than some criterion $1e-4$, say.

A little coding...

- Code the function using the interval utilities,
- Run a simple loop until convergence.
- Try the example with Stata...practical issue - results and box management.

Multivariate inclusion functions

- Multivariate problems are no different.
- We now have an equation system $f_1(x_1, x_2, \ldots, x_n)$, $f_2(x_1, x_2, \ldots, x_n)$, and so on.
- A great inclusion function is the *Taylor* inclusion function. In one dimension, this is:

$$f([x]) \subset f(m) + [f']([x])([x] - m)$$

- n dimensional generalization: f, x $n-$dimensional vectors:

$$\mathbf{f}([\mathbf{x}]) \subset \mathbf{f}(\mathbf{m}) + [\mathbf{J}]([\mathbf{x}])([\mathbf{x}] - \mathbf{m})$$

Converges **quadratically**. There are even better possibilities.

Contracting boxes

The taylor inclusion function leads to the idea of trying to contract boxes equation by equation, and this is how my solver works.

- Essentially, isolates one variable and one equation at a time, and enforces consistency.
- Based on Kearfott (1990); advantage is it avoids computing "everything" for every iteration.
- Can be improved to check for uniqueness within solution boxes.
- Requires defining interval division $[x]/[y]$, which is a little tricky because of the possibility that $[y]$ includes zeros.

Take a look at Stata, and some "uniform" games:

- 2-players
- 3-players
- 4-players

- A solver for games that hopefully you can use.
- A beta version will be up for playing around with soon; I'll let you know!
- A seemingly simple problem that is quite difficult computationally, because of need to solve polynomial systems and enumerate all possible supports. To solve a five player game, there are $2^5 = 32$ strategy profiles to check, one 5 equation system, 10 4 equation systems, etc.
- Coming soon: Stata code using the solver to implement Bajari, Hong, and Ryan (2010).