A New Kind of Fictitious Play

Alan Washburn

Operations Research Department
Naval Postgraduate School
Monterey, California 93943

Received July 2000; revised November 2000; accepted 9 January 2001

Abstract: Gass, Zafra, and Qiu (1996) introduce a modified version of fictitious play for matrix games where only the more successful player's strategy is retained. This paper introduces additional modifications that appear to converge faster. © 2001 John Wiley & Sons, Inc.* Naval Research Logistics 48: 270–280, 2001

Keywords: game; fictitious play

1. BACKGROUND

It has been known since 1928 (von Neumann [11]) that every finite, two-person-zero-sum (TPZS) game has a solution, but practical methods for computing solutions were not available until the advent of digital computers. Once the equivalence between TPZS games and linear programming was established (Dantzig [3]), the simplex method became available as one possibility for solving games. For most games, the simplex method is still the method of choice, but other methods (Brown and von Neumann [2], Brown [1], von Neumann [12]) have also been proposed. These alternative methods are still of interest on account of their potential ability to approximately solve large linear programs, as well as TPZS games. The subject of this paper is a modified version of the Brown–Robinson fictitious play (FP) method (Brown [1]). FP has an intuitively appealing description. The two players, P1 (the maximizing row player) and P2 (the minimizing column player), simply take turns choosing strategies, with each player choosing whatever strategy is optimal against the empirical distribution of strategy choices made so far by the other. Each choice by P1 (P2) establishes a lower (upper) bound on the value of the game, and it is known that the bounds must each approach the value of the game as the number of iterations k approaches infinity (Robinson [8]). Unfortunately, the convergence is very slow. For a game with m rows and n columns, Shapiro [9] shows that the order of convergence is at worst $O(k^{-1/(m+n-2)})$. Szép and Forgó [10] conjecture that convergence is actually $O(k^{-1/2})$. Zafra [14] offered a proof of square-root convergence, but a subtle technical point in the proof has since been raised that has not, as yet, been resolved (Zafra [14]). Square-root convergence is typical of computational experience.

The present work can be thought of as a continuation of the work by Gass, Zafra, and Qiu [6], (referred to hereafter as GZQ) to find a modified version of FP that exhibits faster convergence.

The hoped-for convergence rate is $O(k^{-1})$; convergence at a higher rate seems unlikely since any fictitious play procedure will have to deal with the granularity involved in approximating mixed strategies by fractions whose denominator is the number of iterations.

Several fictitious play algorithms will be described and tested in this article. All involve operations that are naturally and compactly expressed in terms of matrix rows and columns, so the MATLAB language is chosen for representation. MATLAB notation will also be used in the text for consistency, except that transposes will be indicated by a "t" superscript. The only MATLAB idiosyncrasy that the reader needs to be aware of is the use of the colon as a placeholder for "the whole row" or "the whole column," depending on placement. All scripts referenced below can be downloaded from http://diana.or.nps.navy.mil/∼washburn/ModFicPlay/.

2. FIVE FICTITIOUS PLAY PROCEDURES

2.1. Regular Fictitious Play (FP)

Let $\mathbf{A} = (a_{ij})$ be the payoff matrix of a two-person-zero-sum game, let $\mathbf{A}(i, :)$ be the ith row of $\mathbf{A}, i = 1, \ldots, m$, and let $\mathbf{A}(:, j)$ be the jth column of $\mathbf{A}, j = 1, \ldots, n$. We will make use of accumulators U and V for columns and rows, respectively, initially set to be zero vectors. FP begins with P1 making a row choice; to be definite, let P1's initial choice be any row i that achieves $\max_i \min_j(a_{ij})$. V is then augmented by $\mathbf{A}(i, :)$. Whenever it is P2's turn to choose a column, he chooses $j =$ argmin (V), after which U is augmented by $\mathbf{A}(:, j)$. At all moves except the first, P1 chooses $i =$ argmax (U), after which V is augmented by $\mathbf{A}(i, :)$. Iteration k is by definition complete whenever P2 has chosen his kth strategy, at which point U and V are each the accumulation of k vectors. Note that V/k is a vector of payoffs against the empirical mixed strategy that consists of P1's first k choices, and similarly for U/k. The division by k is not necessary for determining the next column choice, and is therefore omitted unless bounds on the game value are required.

The empirical strategies after iteration k provide bounds $\min(V)/k \leq v \leq \max(U)/k$ on the game value v. Each of these local bounds will approach v as k grows large. The convergence is not monotonic, but a monotonic bound can always be found by keeping track of the best local bound found so far. Except for keeping track of bounds, the FP algorithm requires comparison and addition, but neither subtraction, multiplication, nor division. The bounds require division by k, but only two per iteration. The essence of the FP algorithm is that it requires only simple arithmetic and very little memory aside from storing matrix $\mathbf{A}$ itself. These are characteristics that we hope to retain in modifications. The FP algorithm also converges very slowly, which is what we hope to change.

MATLAB script *fp.m* expresses fictitious play as described above. MATLAB script *matgen.m* is used below for generating 50% dense payoff matrices whose nonzero elements are uniformly distributed between -100 and 100 for testing purposes, the same class of matrices used by GZQ as well as Gass and Zafra [5]. The 100×100 matrices generated by *matgen.m* will be called "standard" matrices.

2.2. Modified Fictitious Play (MFP)

Symmetric games by definition have a skew-symmetric payoff matrix; that is, $a_{ij} = -a_{ji}$ for all i, j. The value of a symmetric game is zero, and any mixed strategy optimal for one side is also optimal for the other. An efficient method for computing optimal mixed strategies will take advantage of the fact that the value of the game is already known. The MFP algorithm developed by Zafra [13] does this in a very simple but effective way. After each iteration, the local bounds are compared to see which is closer in absolute value to 0. If the lower bound is closer, then P2 simply adopts P1's strategy from that point on, or vice versa if the upper bound is closer. GZQ show this method to be more effective than FP in symmetric games. We assume here that the strategy comparison is made after every iteration, since GZQ find no reason to do so less frequently.

GZQ also apply their MFP method to general games, but it is first necessary to symmetrize the game A so that its value is 0. The GZQ symmetrization method is that of Gale, Kuhn, and Tucker [4], hereafter GKT. GKT employ the symmetric matrix

$$\mathbf{S} \equiv \begin{bmatrix} 0 & A & -\mathbf{1} \\ -A^t & 0 & \mathbf{1} \\ \mathbf{1}^t & -\mathbf{1}^t & 0 \end{bmatrix},$$

where A^t is the transpose of $\mathbf{A}$ and $\mathbf{1}$ is an appropriately dimensioned column vector of 1's. Let (x, y, λ) solve $\mathbf{S}$, and let $\mu = y_1 + \cdots + y_n$. As long as the value of game A is positive, GKT prove that x/μ and y/μ are optimal strategies for P1 and P2 in $\mathbf{A}$, and that the value of game A is λ/μ. In other words, solution of any game with a positive value is no harder than solution of a larger symmetric game. GZQ combine this theorem with their ability to solve symmetric games by MFP to obtain a method that works on games in general.

Changes of scale and location of the matrix $\mathbf{A}$ are computationally significant when MFP is employed on $\mathbf{S}$. If the scale of the numbers in $\mathbf{A}$ is large compared to 1, then the x and y vectors end up being starved of row and column choices, and most of the probability ends up in λ. GZQ note this, but also note that the GKT theorem remains valid if the 1-vectors in $\mathbf{S}$ are multiplied by any positive constant. Computational tests reveal that the multiplier c should be on the same scale as the numbers in $\mathbf{A}$—GZQ recommend 75% of the distance from the smallest number in $\mathbf{A}$ to the largest. Tests of the resultant procedure show that MFP is superior to FP in terms of iteration count, and on some problems even in terms of cpu time, as the number of iterations becomes large. MATLAB script *mfp.m* implements this method.

The GKT theorem requires that $\mathbf{A}$ have a positive value. GZQ assure that this is the case by simply adding a constant to all the entries of $\mathbf{A}$ to ensure that none of them are negative. The addition is strategically neutral, so the revised payoff matrix has the same optimal strategies as $\mathbf{A}$. However, the addition is not *computationally* neutral, and it is clear that the MFP method suffers if the value of the adjusted $\mathbf{A}$ is unnecessarily large. The best adjustment in practice usually turns out to be one that makes the value of the modified game just barely positive.

MFP will still produce bounds on the game value even if the value of game A is negative, but the bounds do not converge to the game value. The problem is that the rows of $\mathbf{S}$ corresponding to strategies for P1 are almost never used, and in the final analysis one merely has a strategy for P2 that guarantees that the payoff will not be positive (if *all* of the elements of $\mathbf{A}$ were negative, then $\mathbf{S}$ would have a saddle point in the center where both players of $\mathbf{S}$ use only strategies corresponding to the columns of $\mathbf{A}$; i.e., P2). It is tempting to occasionally reverse the sign of c in MFP if necessary to assure that P1's strategies are also used. This impulse is justified by the following generalization of the GKT Theorem:

Generalization of the **GKT THEOREM:**

Let

$$\mathbf{S} = \begin{bmatrix} 0 & A & -C_m \\ -A^t & 0 & C_n \\ C_m^t & -C_n^t & 0 \end{bmatrix},$$

where C_m and C_n are column vectors composed entirely of the constant c, and let (x, y, λ) solve the symmetric game $\mathbf{S}$ for some $c > 0$. That is, suppose

$$\sum_j a_{ij} y_j \leq c\lambda \qquad \forall i, \tag{1}$$

$$\sum_i a_{ij} x_i \geq c\lambda \qquad \forall j, \tag{2}$$

$$c \sum_i x_i \leq c \sum_j y_j, \tag{3}$$

$$\sum_i x_i + \sum_j y_j + \lambda = 1, \tag{4}$$

$$x, y, \lambda \geq 0. \tag{5}$$

Let $X = \sum_i x_i$ and $Y = \sum_j y_j$. Likewise let (x', y', λ') solve $\mathbf{S}$ for some $c' < 0$, with X' and Y' defined similarly to X and Y. Then

1. If $\lambda > 0$, the optimal strategies and value of $\mathbf{A}$ are $(x/Y, y/Y, c\lambda/Y)$.
2. If $\lambda' > 0$, the solution of $\mathbf{A}$ is $(x'/X', y'/X', c'\lambda'/X')$.
3. If $\lambda = 0$ and $\lambda' = 0$, the solution of $\mathbf{A}$ is $(x'/X', y/Y, 0)$.

In all cases, it suffices to solve at most two symmetric games.

PROOF: If $Y = 0$, then $X = 0$ by (3) and (5), $\lambda = 0$ by (2), and $1 = 0$ by (4). This contradiction proves that $Y > 0$. If also $\lambda > 0$, then, upon multiplying each inequality in (1) by x_i and summing, and likewise multiplying each inequality in (2) by y_j and summing, we find that $X \geq (1/\lambda) \sum_i \sum_j a_{ij} x_i y_j \geq Y$. When combined with (3), this implies $X = Y$. Equations (1) and (2) then imply that the value of the game is $c\lambda/Y$, with the optimal strategies being x/Y and y/Y.

Thus far this is the original GKT proof, with the rest of that proof consisting of showing that λ *must* be positive because the value of game A is assumed positive. We proceed differently.

Since c' is negative, it is X' that must be positive, rather than Y'. If also $\lambda' > 0$, then an argument similar to the one in the positive case shows that $Y' = X'$ and that a solution of $\mathbf{A}$ is $(x'/X', y'/X', c'\lambda'/X')$. Thus $\mathbf{A}$ is solved unless both $\lambda = 0$ and $\lambda' = 0$. But if $\lambda' = 0$, then x'/X' still guarantees an average payoff of at least 0 by (2), and if $\lambda = 0$, then y/Y still guarantees an average payoff of at most 0 by (1). Therefore, $\mathbf{A}$ is solved even in this case. □

A direct use of this theorem involves a computational gamble. If the value of game A is suspected to be positive, then clearly $\mathbf{S}$ should first be solved for positive c. If $\lambda > 0$, the negative case will not need to be solved. Similarly c should be made negative if the value of game A is suspected to be negative. An alternative, of course, is to simply add a large constant to $\mathbf{A}$ to assure that its value is positive, thereby avoiding the gamble. As mentioned above, however, MFP is most efficient when the value of game A is near zero. The revised GKT theorem permits MFP to dynamically adjust $\mathbf{A}$ with this in mind.

2. 3. Dynamic Modified Fictitious Play (DMFP)

The idea of dynamic versions of MFP was first suggested by GZQ. Limsakul [7] (the author's student at the time) retains the GZQ formula for c and uses a dynamic procedure where the current best lower bound on the value of game A is subtracted from every element, thus assuring that the modified matrix has a small but nonnegative value. He reports encouraging results for his dynamic procedure in comparison with MFP. Our version (DMFP) instead sets c to be the currently estimated game value v, estimated by simply averaging the current upper and lower bounds. The matrix $\mathbf{A}$ is also in effect adjusted before each iteration by subtracting v from every element of $\mathbf{A}$.

If c were any fixed constant with the same sign as the game value, DMFP would amount to applying FP to a symmetric game, Robinson's FP convergence proof would apply, and the original game $\mathbf{A}$ would also be approximately solved according to the revised GKT theorem. In fact there is no convergence theorem for DMFP because c is not constant, but still DMFP seems to be reliable. In other words, the DMFP situation is very much like the FP situation when Brown first proposed it in 1951 [1].

Figure 1 shows comparative performance of FP, MFP, and DMFP on the first 100×100 standard game generated by *matgen.m* after the random number generator is reset. Notable features are that DMFP is a substantial improvement over MFP, and that FP is superior to both MFP and DMFP

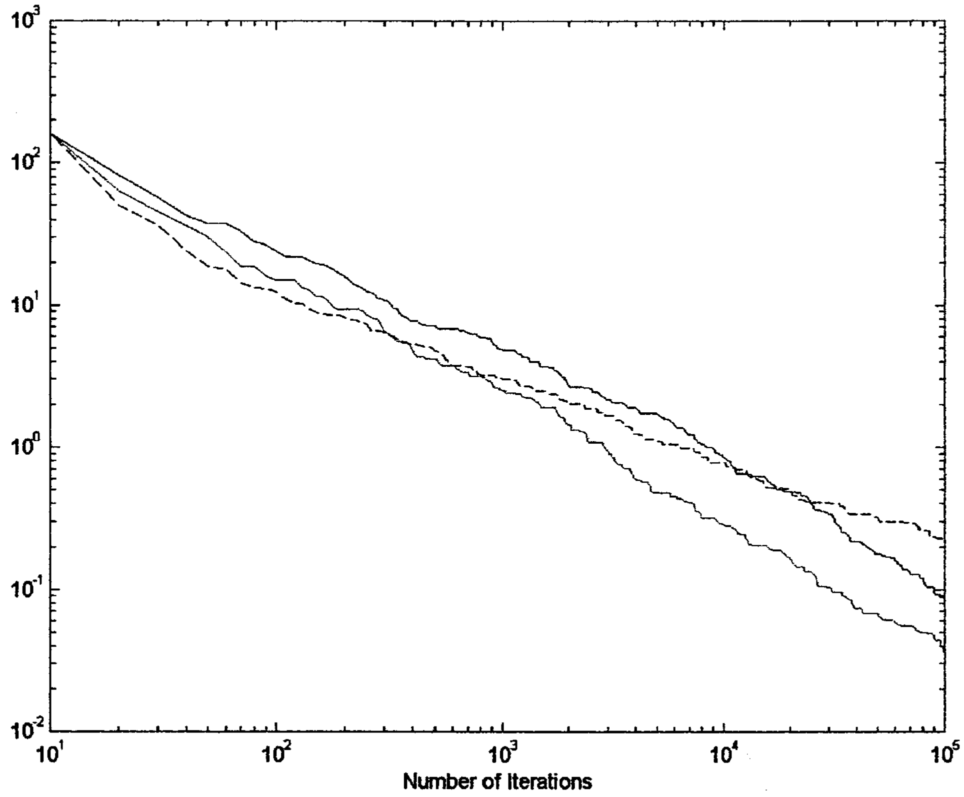

Figure 1. Gaps for procedures MFP (upper, solid), FP (dashed), and DMFP (lower, solid) on a standard 100×100 problem.

when there are only a few hundred iterations. FP has an advantage in incrementing strategies for *both* P1 and P2 on every iteration, but the advantage is finally given up as the number of iterations becomes large. Results are remarkably similar for all standard games (see Section 3).

If the game value were for some reason known to be some specific value v, it would be natural to simply set $c \equiv v$ in (1)–(5) to discover the optimal strategies for P1 and P2. It is easy to show that one solution of (1)–(5) in that case is where $X = Y = \lambda = 1/3$. This is also the computational experience with MFP in most cases; that is, neither X nor Y is incremented in about one third of the iterations. These iterations are in a sense wasted, since they do not contribute directly to the estimation of optimal strategies for P1 and P2.

In the special case where $v = 0$, which is what DMFP aims for by adjustment, $\mathbf{S}$ has a saddle point in the lower right-hand corner. In that case (1)–(5) has a family of solutions where λ is any number between 0 and 1. In practice, DMFP seems to converge to solutions where λ is small, so the number of wasted iterations is not significant. But why waste *any* iterations when the problem can be avoided by simply omitting the last row and column of $\mathbf{S}$? GKT inserted them for a purpose, but it turns out that they are not needed in solving a game by fictitious play.

2. 3. 1. A Minimal Symmetrization

Consider the symmetrization that omits the last row and column of $\mathbf{S}$:

$$\mathbf{B} = \begin{bmatrix} 0 & A \\ -A^t & 0 \end{bmatrix},$$

with $m + n$ rows and $m + n$ columns, and let (x, y) be an optimal mixed strategy for $\mathbf{B}$. Since the value of $\mathbf{B}$ is zero, necessarily

$$\sum_j a_{ij} y_j \le 0 \qquad \forall i, \tag{6}$$

$$\sum_i a_{ij} x_i \ge 0 \qquad \forall j, \tag{7}$$

$$\sum_i x_i + \sum_j y_j = 1, \tag{8}$$

$$x, y \ge 0. \tag{9}$$

As before, let $X = \sum_i x_i$ and $Y = \sum_j y_j$. If $X > 0$, then x/X is a mixed strategy for P1 that guarantees a nonnegative payoff by (7). Likewise y/Y guarantees a nonpositive payoff for P2 by (8) if $Y > 0$. Now, at least one of X and Y must be positive. Should it happen that *both* are positive, then clearly the value of game A is zero, and a solution is at hand. But this is unlikely to be the case. If the value of game A is positive, for example, then it is not hard to show that Y has to be 0, in which case all that would be learned from solving $\mathbf{B}$ would be that the value of game A is in fact positive, together with a strategy x that that enforces a positive payoff. The limited utility of this result explains why the GKT symmetrization employs an extra column. At the cost of one more column, the solution of any game with positive value is implicit in the solution of $\mathbf{S}$.

However, the minimal symmetrization does have charms aside from its slightly smaller size. If the value of game A happens to be known for some reason, it can be subtracted from all the elements of $\mathbf{A}$ to obtain a game with zero value before solving $\mathbf{B}$. In an iterative procedure where the value of game A is dynamically estimated rather than known, one might expect the value of the (adjusted) $\mathbf{A}$ to be sometimes slightly positive and sometimes slightly negative, so that neither x nor y will be starved of strategy choices. This is what we have in mind in the procedure MFP2.

2. 4. A New Modified Fictitious Play Procedure (MFP2)

Let matrix $\mathbf{B}$ be the minimally symmetrized version of $\mathbf{A}$. We will refer to the players of $\mathbf{B}$ as "agents" to distinguish them from the players P1 and P2 of the original game $\mathbf{A}$. Although the two agents can be viewed as applying MFP to a dynamically adjusted version of $\mathbf{B}$, it is computationally efficient and intuitively useful to describe all of their operations in terms of the rows and columns of $\mathbf{A}$. MATLAB script *mfp2.m* makes no explicit reference to matrix $\mathbf{B}$.

A strategy for each agent consists of a strategy x for player 1 *and* a strategy y for player 2, where now vectors x and y are assumed to hold counts of the number of times each row or column of $\mathbf{A}$ have been chosen in previous iterations of MFP2. As usual, we define $X = \sum_i x_i$ and $Y = \sum_j y_j$. Since each iteration of MFP2 chooses either a row of $\mathbf{A}$ or a column of $\mathbf{A}$, $X + Y$ is the number of MFP2 iterations so far. At the beginning of each iteration, the two agents have the same strategy (x, y) and one agent (agent a, say) must determine the next row or column to choose. Let $U(y)$ be the sum of all the columns of $\mathbf{A}$ represented in y, and likewise let $V(x)$ be the sum of all the rows in x. Thus $U(y)$ is the sum of Y columns and $V(x)$ is the sum of X rows. Let v be the current estimate of the value of game A, the average of the lower and upper bounds on the game value. Define

$$U^*(y) \equiv \max_i U_i(y) - vY,$$

$$I(y) \equiv \arg\max_i U_i(y) - vY,$$

$$V^*(x) \equiv vX - \min_j V_j(x),$$

$$J(x) \equiv vX - \arg\min_j V_j(x).$$

Agent a selects row $I(y)$ if $U^*(y)$ is larger than $V^*(x)$, or otherwise column $J(x)$. This is exactly the choice that agent a would make in fictitious play if agent b were using the strategy (x, y), and if the payoff matrix $\mathbf{A}$ were adjusted by subtracting v from every element. The ratio $U^*(y)/Y$ is the best "excess over v" that agent a can find by choosing a row against (x, y), with a similar interpretation for $V^*(x)/X$. Thus agent a essentially compares the row excess with the column excess and chooses the larger. Both $U^*(y)$ and $V^*(x)$ must be nonnegative if v is the exact game value, so at least one must be nonnegative even if v is not. The larger of the two (agent a's "gain") will therefore always be nonnegative.

Let (x', y') be (x, y) augmented by player 1's choice, and let agent b's choice be the same as agent a's except that (x, y) is replaced by (x', y'). Agent b will choose $I(y')$ if $U^*(y')$ is larger than $V^*(x')$, else $J(x')$. Let (x'', y'') be (x, y) augmented by agent b's choice. Then (x', y') will be adopted if agent 1 has the larger gain, else (x'', y''). The adopted strategy then becomes (x, y) for the next iteration.

One point should be emphasized about the operation of MFP2. Although agent b selects the row or column that is the best response to (x', y'), he adds that row or column to (x, y) [not to (x', y')] to get (x'', y''). Therefore, the effect of each iteration is always to add a single row or column to the input strategy (x, y). Thus, while regular FP adds both a row *and* a column of $\mathbf{A}$

on each iteration, MFP2 adds *either* a row or a column. MFP (or DMFP) adds either a row or a column or nothing, the last possibility corresponding to a choice of the last row or column of $\mathbf{S}$. The three procedures thus have different levels of aggressiveness, if one thinks of aggressiveness as the tendency to achieve efficiency by adding lots of rows and columns quickly. This is an important issue because *any* fictitious play procedure can only get rid of early bad choices by later swamping them with good ones.

Early versions of MFP2 were haunted by the possibility that either x or y would be so starved of choices that the corresponding lower or upper bound would not converge to the game value. Various experiments were tried to prevent this. The most successful simply made an adjustment δ_k for the kth iteration, with δ_k being added to the elements of $\mathbf{A}$ if $X < Y$ or subtracted if $Y < X$. If $\delta_k =$ (positive const)/k, this will prevent starvation of either x or y because the harmonic series sums to infinity (the accumulated corrections would eventually make all elements or the adjusted $\mathbf{A}$ be positive if $X < Y$ consistently, or vice versa if $Y > X$ consistently). However, MFP2 does not incorporate this adjustment because it simply doesn't seem to be necessary; the natural fluctuations in the estimated game value fulfill the same purpose.

MFP2 performs similarly to DMFP in terms of gap versus iteration count. A plot similar to Figure 1 would only reveal that the two curves nearly overlie one another. However, MFP2 has an advantage in terms of computation load per iteration (see Section 3).

2. 5. The MFPN Fictitious Play Procedure

The description of MFP2 given above permits an easy generalization of the number of agents. If there were only one agent, then (x', y') would be adopted automatically. If there were three, then the third agent would make his choice based on (x'', y''), and there would be three gains to compare, etc. The same logic works with N agents as long as $N \geq 1$. We will use the notational convenience of referring to MFPN with three agents as MFP3, etc. With more agents, the adopted row or column is presumably less likely to be a mistake, but of course more computation is required per iteration.

Is MFPN really "fictitious play?" It is hard to justify the term unless $N = 2$. MFPN is probably better regarded as a procedure for solving games that is based primarily on comparison and addition. A close examination of MATLAB script *mfpn.m* will show that each iteration of MFPN requires no subtractions, one division (by an integer in computing bounds) and a few multiplications (in the process of updating v). The basic idea in MFPN, as in all forms of fictitious play, is to use a simple procedure that requires very little memory.

3. COMPUTATIONAL TESTS

This section provides some computational comparisons of the five versions of fictitious play described above. All procedures except for FP begin by including both the maxmin row and the minmax column in the strategy vector, so the gap will be zero for any game having a saddle point. Tests are therefore performed on games where the solution can be expected to involve mixed strategies, primarily standard games.

Table 1 shows the results for the first 30 100×100 standard games generated by *matgen.m,* 100,000 iterations each. There is very little variability among the 30 games, as reflected in the small sample standard deviations of the gap.

MATLAB provides a measure of floating point operations ("flops") that is converted to Megaflops by dividing by 10^6 and averaged over 30 games in Table 1. Each Megaflop takes about a second on the 450 MHZ WINTEL machine used for these computations, so DMFP takes

Table 1. Each procedure solves 30 standard games for 100,000 iterations.[a]

	FP	MFP	DMFP	MFP2	MFP3
Average gap	0.200	0.094	0.035	0.034	0.029
Std. deviation	0.017	0.010	0.003	0.004	0.003
Megaflops	20.5	61.5	128.6	15.2	32.1
Score	N/A	5.78	4.50	0.52	0.93

[a] MFP2 has the best combination of small Average Gap and small Megaflops, as reflected in the Score.

about a minute per game, etc. FP makes 2 vector additions per iteration, whereas MFP2 makes only about 1.5 by being careful not to make the second one when it is not necessary. MFPN makes N vector additions per iteration when there are N agents, and therefore 3 when $N = 3$. These observations explain the relative sizes of the Megaflops measure for FP, MFP2, and MFP3, each of which deals directly with rows and columns of the game matrix $\mathbf{A}$ in the MATLAB implementation. MFP and DMFP manipulate the rows and columns of larger symmetric matrices, and pay for doing so in the Megaflops measure. DMFP and MFP2 provide almost identical gaps in Table 1, but MFP2 does so with much less effort.

Since all procedures except FP have the gap being approximately inversely proportional to the number of iterations when the number of iterations is large (see Fig. 1), a scalar measure Score $\equiv$ (average Gap)(Megaflops) is given for each procedure. The procedure with the smallest Score will have the smallest gap for any fixed but large number of Megaflops. It is evident that the best choice of procedure for small gaps is MFP2. MFP3 achieves a slightly smaller gap per iteration, but with about twice as many Megaflops.

Table 2 shows results for a similar experiment where the games are 50×200. The Score and relative results are similar to Table 1. Although the number of elements in $\mathbf{A}$ is the same in each table, the number of Megaflops is approximately $(200 + 50)/(100 + 100)$ times as large in Table 2. In other words, computational effort is proportional to $m + n$. This is reasonable for FP because a row and a column must be added in each iteration, and reasonable for the other procedures because the dimension of the underlying symmetric matrix is close to $m + n$. The gap decreases somewhat in Table 2 compared to Table 1, the beginning of a trend toward zero for games consisting of a single row.

It was mentioned earlier that λ tends to be about 1/3 in MFP, but to be smaller in DMFP. In fact, λ is smaller than 0.01 in all of the games solved by DMFP in Tables 1 and 2. MFP2 can be thought of as a version of DMFP where λ is forced to be zero, so the closeness of the gaps for DMFP and MFP2 in both tables is not surprising.

Figure 2 shows the results of an experiment where games of increasingly large size are solved to increasingly small gaps by MFP2. The curves are all slightly concave, with the slope approaching

Table 2. Solution of 30 50×200 games for 100,000 iterations.[a]

	FP	MFP	DMFP	MFP2	MFP3
Average gap	0.162	0.079	0.030	0.029	0.024
Std. deviation	0.017	0.010	0.003	0.004	0.003
Megaflops	25.5	76.5	160	18.8	40.9
Score	N/A	6.04	4.80	0.55	0.98

[a] Results are similar to Table 1.

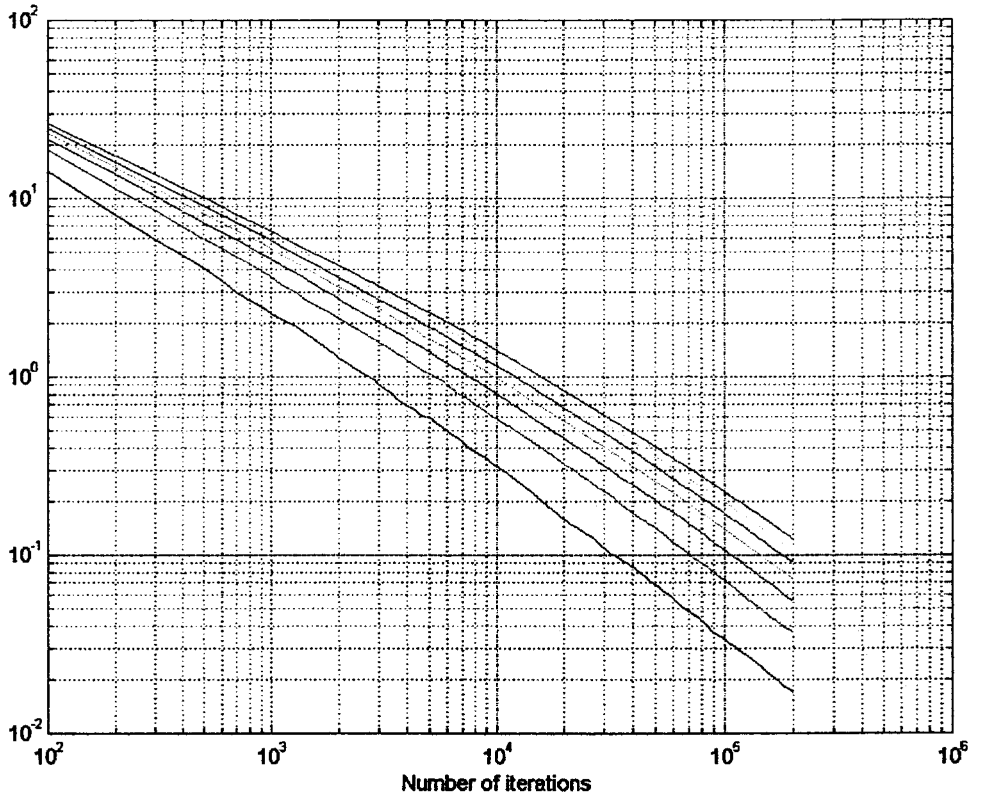

Figure 2. MFP2 gap averaged over 30 standard square games and plotted versus the number of iterations. The games range in size from 100 × 100 (bottom curve) to 700 × 700 (top curve).

the hoped-for -1 as the number of iterations becomes large. The gap increases with the dimension of the game, but less than proportionally.

These results hold promise for the efficient solution of large games or linear programming problems. Gass and Zafra [5] describe some encouraging experiments where MFP is used to determine a nearly optimal starting basis for the simplex algorithm. The idea of using fictitious play as a first stage may become more attractive as more efficient versions are developed.

4. SUMMARY

Two new forms of fictitious play are introduced. The first (DMFP) relies on the modified GKT theorem to permit a dynamic version of MFP where the game matrix **A** is not required to have a positive value. The second (MFP2) employs a different symmetrization than MFP and DMFP. Among known fictitious play procedures, MFP2 appears to be the most efficient way to approximately solve TPZS games, particularly when the approximation needs to be accurate. MFP2 generalizes easily to utilize an arbitrary number of agents, but the computationally optimal number of agents appears to be two.

ACKNOWLEDGMENTS

Helpful discussions with both Piya Limsakul and James Eagle are acknowledged.

REFERENCES

[1] G.W. Brown, ‘‘Iterative solution of games by fictitious play,’’ Activity analysis of production and allocation, T.C. Koopmans (Editor), Wiley, New York, 1951, pp. 374–376.

[2] G.W. Brown and J. von Neumann, ‘‘Solutions of games by differential equations,’’ Contributions to the theory of games, H.W. Kuhn and A.W. Tucker (Editors), Annals of Mathematics Study No. 24, Princeton University Press, Princeton, NJ, 1950, Vol. 1, pp. 73–79.

[3] G. Dantzig, ‘‘A proof of the equivalence of the programming problem and the game problem,’’ Activity analysis of production and allocation, T.C. Koopmans (Editor), Wiley, New York, 1951, pp. 339–347.

[4] D. Gale, H.W. Kuhn, and A.W. Tucker, ‘‘On symmetric games,’’ Contributions to the theory of games, H.W. Kuhn and A.W. Tucker (Editors), Annals of Mathematics Study No. 24, Princeton University Press, Princeton, NJ, 1950, Vol. 1, pp. 81–87.

[5] S.I. Gass and P.M.R. Zafra, Modified fictitious play for solving matrix games and linear-programming problems, Comput Oper Res 22 (1995), 893–903.

[6] S.I. Gass, P.M.R. Zafra, and Z. Qiu, Modified fictitious play, Nav Res Logistics 43 (1996), 955–970.

[7] P. Limsakul, Iterative algorithms for two-person zero-sum games, Master’s thesis, Naval Postgraduate School, 1999.

[8] J. Robinson, An iterative method of solving a game, Ann Math 54 (1951), 296–301.

[9] H.N. Shapiro, Note on a computation method in the theory of games, Commun Pure Appl Math 11 (1958), 587–593.

[10] J. Szép and F. Forgó, Introduction to the theory of games, Reidel, Dordrecht, 1985.

[11] J. von Neumann, Zur Theorie der Gesellschaftsspiele, Math Ann 100 (1928), 295–320.

[12] J. von Neumann, A numerical method to determine optimum strategy, Nav Res Logistics Quart 1 (1954), 109–115.

[13] P. Zafra, Theoretical and computational advances in the fictitious play method and some applications to the solution of linear programming problems, Ph.D. dissertation, University of Maryland, College Park, May 1993.

[14] P. Zafra, personal communication, 1999.