Best Experienced Payoff Dynamics and Cooperation in the Centipede Game: Online Appendix

William H. Sandholm[*], Segismundo S. Izquierdo[†], and Luis R. Izquierdo[‡]

April 24, 2019

Contents

[*]Department of Economics, University of Wisconsin.
[†]Department of Industrial Organization, Universidad de Valladolid.
[‡]Department of Civil Engineering, Universidad de Burgos.

I. Exact and numerical calculation in *Mathematica*

In this section we describe the built-in *Mathematica* functions we use to prove exact (analytical) results and to obtain numerical evaluations of exact expressions.

I.1 Algebraic numbers and solutions to polynomial equations

To obtain our analytical results, we take advantage of *Mathematica*'s ability to perform exact computations using algebraic numbers. As described in Strzeboński (1996, 1997), *Mathematica* represents algebraic numbers using `Root` objects, with `Root[`*poly*`,` *k*`]` designating one of the roots of the minimal polynomial *poly*. The index *k* is used to single out a particular root of *poly*, with the lowest indices referring to the real roots of *poly* in increasing order, and the higher indices referring to the complex roots in a more complicated way. `Root` objects also contain a hidden third element that specifies an *isolating set* for the root, meaning a set containing the root of *poly* in question and no others.

The forms of isolating sets depend on whether roots are isolated using arbitrary-precision floating point methods or exact methods. If *Mathematica*'s default settings are used, then roots are isolated using arbitrary-precision floating point methods based on the Jenkins-Traub algorithm (Jenkins (1969), Jenkins and Traub (1970a,b)), the workhorse numerical algorithm for this purpose. While in theory this algorithm always isolates all real and complex roots of *poly* in disjoint disks in the complex plane, flawless implementation of the algorithm is difficult; see Strzeboński (1997, p. 649).

If we instead use the setting

```
SetOptions[Root,ExactRootIsolation->True]
```

then *Mathematica* isolates roots using exact methods—that is, methods that only use rational number calculations. Real roots of polynomials are isolated in disjoint intervals using the Vincent-Akritas-Strzeboński method, which is based on Descartes' rule of signs and a classic theorem of Vincent; see Akritas et al. (1994) and Akritas (2010). Complex roots are isolated in rectangles using the Collins and Krandick (1992) method.

Exact roots of univariate polynomials (and much else) can be computed using the *Mathematica* function `Reduce`. When computing the exact rest points of BEP dynamics, we apply `Reduce` to the output of the function `GroebnerBasis`, described next.

I.2 Algorithms from computational algebra

The *Mathematica* function `GroebnerBasis` is an implementation of a proprietary variation of the algorithm of Buchberger (1965, 1970).[1] Choosing the option `Method -> Buchberger` causes *Mathematica* to use the original Buchberger algorithm, which runs considerably more slowly than the default algorithm; however, there was only one case in which the default algorithm produced a Gröbner basis and the Buchberger algorithm failed to terminate.

[1] An up-to-date presentation of Gröbner basis algorithms, including many improvements on Buchberger's algorithm, can be found in Cox et al. (2015).

The *Mathematica* function `CylindricalDecomposition` implements the Collins (1975) cylindrical algebraic decomposition algorithm with various improvements.[2] If this function is run in its default mode, it makes use of arbitrary-precision arithmetic. To force *Mathematica* to work with algebraic numbers, one uses the following settings:

```
SetOptions[Root,ExactRootIsolation->True]
SetSystemOptions["InequalitySolvingOptions"->"CADDefaultPrecision"->Infinity]
```

Unfortunately, these settings cause `CylindricalDecomposition` to run extremely slowly, and in the case of BEP dynamics in Centipede it only generates a result in cases with 2 dimensions and, for some specifications of the dynamics, 3 dimensions. Even if arbitrary-precision arithmetic is permitted, the function generates a result for all BEP dynamics in cases with dimension 2 or 3, but not for higher dimensions.

I.3 Numerical evaluation and precision tracking

When *Mathematica* performs calculations using arbitrary-precision numbers x, it keeps track of the digits whose correctness it views as guaranteed. `Precision[`x`]` reports the number of correct base 10 significant digits of x: for instance, if $x = d_0.d_1d_2d_3d_4\ldots \times 10^k$, the precision is the number of the correct digits in $d_0.d_1d_2d_3d_4\ldots$ `Accuracy[`x`]` is the number of correct base 10 digits of x to the right of the decimal point. Exact numbers in *Mathematica* (e.g., integers, rational numbers, and algebraic numbers) have `Precision` equal to ∞.

To perform certain parts of our analysis (in particular, checking that an eigenvalue of a derivative matrix has negative real part), we need to numerically evaluate exact numbers and expressions. We do so using the *Mathematica* function `N`. `N[`*expr*`,` n`]` evaluates *expr* as an arbitrary-precision number at guaranteed precision n. When *Mathematica* performs computations using arbitrary-precision numbers, it maintains precision and accuracy guarantees, the values of which can be accessed using the `Precision` and `Accuracy` functions.

While in principle *Mathematica*'s precision tracking should not make mistakes, there are at least two reasons for exercising caution when using it in proofs. First, *Mathematica*'s precision tracking is not based on *interval arithmetic*, which represents real and complex numbers using exact intervals (in $\mathbb{R}$) and rectangles (in $\mathbb{C}$) that contain the numbers in question, and which relies on theorems that define rules for performing arithmetic and other mathematical operations on these intervals and rectangles that maintain containment guarantees (Alefeld and Herzberger (1983), Tucker (2011)). Instead, *Mathematica*'s precision bounds are sometimes obtained using faster methods of the Jenkins-Traub variety (see Section I.1), which work correctly in theory but which are difficult to implement perfectly. Second, *Mathematica*'s precision tracking is a black box: the specific algorithms it employs are proprietary.

We contend with these issues by restricting our use of *Mathematica*'s numerical evaluation and precision tracking to a few clearly delineated cases: the evaluation of algebraic numbers, and the basic arithmetic operations of addition, subtraction, multiplication, and

[2] See `reference.wolfram.com/language/tutorial/ComplexPolynomialSystems.html` for details.

division. In particular, we do not use *Mathematica* for precision tracking in the computation of matrix inverses or the solution of linear systems, operations for which interval arithmetic does not generally provide clean answers (Alefeld and Herzberger (1983)). While one could insist that interval arithmetic be used for all non-exact calculations, we chose not to do so.

II. The `BEP_Centipede.nb` notebook

In this section we describe the main functions from the `BEP_Centipede.nb` notebook, which contains all of the procedures we use to analyze BEP dynamics. Section II.1 describes functions used to prove analytical results, and Section II.2 describes the functions used in numerical analyses and in approximations with error bounds (cf. Appendix C). More details about the use of these functions are provided in the `BEP_Centipede.nb` notebook itself. Section II.3 explains the algorithms used to compute numerical values of rest points of the dynamics and eigenvalues of their derivative matrices.

Unless stated otherwise, the functions described below take a test-set rule $\tau \in \{\tau^{\text{all}}, \tau^{\text{two}}, \tau^{\text{adj}}\}$, a tie-breaking rule $\beta \in \{\beta^{\min}, \beta^{\text{stick}}, \beta^{\text{unif}}\}$ and a length d of the Centipede game as parameters. All functions besides the last three are for BEP dynamics with number of trials $\kappa = 1$. The `BEP_Centipede.nb` notebook includes examples of the use of each of the functions.

II.1 Exact analysis

The functions for exact analysis of BEP dynamics in Centipede are as follows:

`ExactRestPoints` Uses `GroebnerBasis` and `Reduce` to compute the exact rest points of the dynamic.

`InstabilityOfVertexRestPoint` Conducts an analysis of the local stability of the vertex rest point $\xi^\dagger$. To do this, the function computes the derivative matrix $D\mathcal{V}(\xi^\dagger)$ of the dynamic and the eigenvalues and eigenvectors of $DV(\xi^\dagger)$, where $V\colon \text{aff}(\Xi) \to T\Xi$ (see Appendix A). Finally, the function reports whether one can conclude that $\xi^\dagger$ is unstable. The function was not used explicitly in our analysis. Instead, we used it to determine the form of the derivative matrix, eigenvalues, and eigenvectors for arbitrary values of d.

`LocalStabilityOfInteriorRestPoint` Conducts an analysis of the local stability of the interior rest point ξ^*. To do this, the function computes a rational approximation ξ of the exact interior rest point ξ^*. The function then evaluates the eigenvalues of $DV(\xi)$, evaluates a version of the perturbation bound from Proposition C.1, and reports whether one can conclude that ξ^* is asymptotically stable.

`GlobalStabilityOfInteriorRestPoint` Conducts an analysis of the global stability of the interior rest point ξ^*. To do this, the function uses `CylindricalDecomposition` to determine whether the relevant Lyapunov function (see Section 3.3) is a strict Lyapunov function for the interior rest point ξ^* on domain $\Xi \setminus \{\xi^\dagger\}$.

II.2 Numerical analysis

The following functions from the `BEP_Centipede.nb` are used for numerical analysis and as subroutines for `LocalStabilityOfInteriorRestPoint`.

`FloatingPointApproximateRestPoint` Computes a floating point approximation of the stable interior rest point of the BEP dynamic. See Section II.3 for details.

`RationalApproximateRestPoint` Computes a rational approximation of the stable interior rest point of the BEP dynamic. See Section II.3 for details.

`EigenvaluesAtRationalApproximateRestPoint` Computes the exact eigenvalues of $DV(\xi)$, where ξ is the rational approximation to the interior rest point obtained from a call to `RationalApproximateRestPoint`. See Section II.3 for details.

`NEigenvaluesAtRationalApproximateRestPoint` Computes the eigenvalues of $DV(\tilde{\xi})$ using arbitrary-precision arithmetic, where $\tilde{\xi}$ is a 16-digit precision approximation to the rational point computed using `RationalApproximateRestPoint`. See Section II.3 for details.

`NumericalGlobalStabilityOfInteriorRestPointLyapunov` Evaluates the time derivative $\dot{\Lambda}(\xi) = \nabla\Lambda(\xi)'V(\xi)$ at a floating-point approximation Λ of the appropriate candidate Lyapunov function L for the interior rest point ξ^*, reporting instances in which the time derivative is not negative should any exist. The (presumably large number of) states ξ at which to evaluate $\dot{\Lambda}(\xi)$ is chosen by the user.

`NumericalGlobalStabilityOfInteriorRestPointNDSolve` Computes numerical solutions to the BEP dynamic from initial conditions provided by the user, and reports whether any of these numerical solutions fails to converge to a neighborhood of the interior rest point ξ^*.

`NDSolveMeanDynamics` Uses *Mathematica*'s `NDSolve` function to compute a numerical solution to the BEP dynamic from an initial condition provided by the user. The solution is computed until the time at which the norm of the law of motion is sufficiently small, where what constitutes sufficiently small can be chosen by the user. The function also graphs the components of the state as a function of time, and reports the terminal point and the time at which this point is reached.

`FloatingPointApproximateRestPointTestAllMinIfTieManyTrials` Uses *Mathematica*'s `FindRoot` function to compute a floating point approximation of a rest point of the BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamic, where the number of trials κ is specified by the user. The function returns only one rest point. When there is more than one rest point, which one is computed depends strongly on the initial condition given to the function as an input. This function was used to produce Figures 3 and 4 in the main paper and to compute the saddle points shown in Table 5 below.

`NDSolveMeanDynamicsTestAllMinIfTieManyTrials` Uses *Mathematica*'s `NDSolve` function to compute a numerical solution of the BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamic, where the number of trials κ and the initial condition of the solution are specified by the user. The solution is computed until the time at which the norm of the law of motion is sufficiently small, where what constitutes sufficiently small can be chosen by the user. The function also graphs the components of the state as a function of time, and reports the terminal point and the time at which this point is reached. The function was used in producing Figure 5.

`EstimateSizeOfBasinOfAttractionOfVertexTestAllMinIfTieManyTrials` Provides an estimate of the size of the basin of attraction of the vertex rest point $\xi^\dagger$ under the BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamic. To do so, it discretizes the set of population states Ξ into a grid whose mesh is chosen by the user, and solves the dynamic with these grid points as initial conditions using *Mathematica*'s `NDSolve` function. It returns the set of initial conditions from which the solution converges to $\xi^\dagger$, and the set of all their neighbors in the grid. See Section IV for details.

II.3 More on computation of approximate rest points and eigenvalues

The `BEP_Centipede.nb` notebook computes approximate rest points of BEP($\tau, 1, \beta$) dynamics using the Euler method: $\{\xi_t\}_{t=0}^{T}$ is computed starting from an initial condition ξ_0 by iteratively applying

$$(1) \qquad \xi_{t+1} = \xi_t + h\,\mathcal{V}(\xi_t),$$

where $\mathcal{V}\colon \mathbb{R}^s \to \mathbb{R}^s$ is the (extended) law of motion of the dynamics and h is the step size of the algorithm. This algorithm is run in two sequential stages, to be described next.

When one of the first two `FloatingPointApproximateRestPoint...` functions from Section II.2 is called, algorithm (1) is run using IEEE 754 Standard double-precision floating-point arithmetic. The step size of the algorithm is set to $h = 2^{-4}$ by default, and the initial condition is $\xi_0 = (x_0, y_0) \in \Xi = (X, Y)$, where x_0 and y_0 are the barycenters of simplices X and Y by default. Several thousand iterations of (1) are run, and the output of each iteration is projected onto Ξ to minimize the accumulation of roundoff errors from the floating-point calculation.

The floating-point numbers obtained in this way are very close to the exact quantities they approximate, but their digits (i.e., the values of the d_i in $x = d_0.d_1d_2d_3d_4\ldots \times 10^k$) may all be wrong, especially in small numbers, since many of the exact numbers we aim to approximate lie outside the range of IEEE 754 double-precision.[3]

To address this issue, the function `RationalApproximateRestPoint` begins with a call to `FloatingPointApproximateRestPoint`, and then uses the output of this procedure to create the initial condition for a second stage that employs rational arithmetic. This initial condition is the rational point in Ξ that lies closest to the floating-point output of the

[3]For example, note that the IEEE 754 double-precision representation of numbers such as 3.78×10^{-681} and 2.18×10^{-20413} (both of which appear in Table 1 below) is 0, since both numbers are well below $2^{-1074} \approx 4.94 \times 10^{-324}$, which is the smallest positive IEEE 754 double-precision number.

first stage. The step size h is set to 1 by default in the second stage, since overshooting is no longer a problem in the neighborhood of the exact rest point. Increment (1) is executed repeatedly using rational arithmetic until it locates a rational point ξ^*_T that is an approximate fixed point of (1), in the sense that ξ_T and $\xi_{T+1} = \xi_T + \mathcal{V}(\xi_T)$ agree with 6 digits of precision for numbers greater or equal to 10^{-4}, or 3 digits of precision for smaller numbers. This agrees with the format we use to report rest points in Section III.

`NEigenvaluesAtRationalApproximateRestPoint` computes the eigenvalues of $DV(\tilde{\xi})$ using arbitrary-precision arithmetic, where $\tilde{\xi}$ is a 16-digit precision approximation to the rational point computed by calling `RationalApproximateRestPoint`. The use of arbitrary precision allows us to keep track of the precision of the computed eigenvalues. Proposition C.1 provides a bound on the distances between the eigenvalues of $DV(\xi)$ and the eigenvalues of $DV(\xi^*)$. In Section III, the reported eigenvalues, which are arbitrary-precision approximations to the (algebraic-valued) eigenvalues of $DV(\xi)$, are shown with 5 digits of precision for numbers greater or equal to 1, 4 digits of precision for numbers greater or equal to 10^{-2}, and 3 digits of precision for smaller numbers.

III. Numerical evaluation of the interior rest point

Table 1 presents approximate components of the unique interior rest point of the BEP($\tau^{\text{all}}, 1, \beta^{\text{min}}$) dynamic in Centipede games of lengths up to $d = 20$.

Table 2 shows approximate eigenvalues of the derivative matrix $DV(\xi^*)$ at the interior rest point ξ^* of BEP($\tau^{\text{all}}, 1, \beta^{\text{min}}$) dynamics in Centipede games of lengths up to $d = 20$.

p	[6]	[5]	[4]	[3]	[2]	[1]	[0]
3	-	-	-	-	-	.618034	.381966
4	-	-	-	-	.113625	.501712	.384663
5	-	-	-	-	.113493	.501849	.384658
6	-	-	-	3.12×10^{-9}	.113493	.501849	.384658
7	-	-	-	3.12×10^{-9}	.113493	.501849	.384658
8	-	-	8.23×10^{-137}	3.12×10^{-9}	.113493	.501849	.384658
9	-	-	8.23×10^{-137}	3.12×10^{-9}	.113493	.501849	.384658
10	-	7.75×10^{-3403}	8.23×10^{-137}	3.12×10^{-9}	.113493	.501849	.384658
11	-	7.75×10^{-3403}	8.23×10^{-137}	3.12×10^{-9}	.113493	.501849	.384658
12	1.06×10^{-122476}	7.75×10^{-3403}	8.23×10^{-137}	3.12×10^{-9}	.113493	.501849	.384658
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
20	1.06×10^{-122476}	7.75×10^{-3403}	8.23×10^{-137}	3.12×10^{-9}	.113493	.501849	.384658

q	[6]	[5]	[4]	[3]	[2]	[1]	[0]
3	-	-	-	-	.381966	.381966	.236068
4	-	-	-	-	.337084	.419741	.243175
5	-	-	-	.001462	.335672	.419706	.243160
6	-	-	-	.001462	.335672	.419706	.243160
7	-	-	9.53×10^{-35}	.001462	.335672	.419706	.243160
8	-	-	9.53×10^{-35}	.001462	.335672	.419706	.243160
9	-	3.78×10^{-681}	9.53×10^{-35}	.001462	.335672	.419706	.243160
10	-	3.78×10^{-681}	9.53×10^{-35}	.001462	.335672	.419706	.243160
11	2.18×10^{-20413}	3.78×10^{-681}	9.53×10^{-35}	.001462	.335672	.419706	.243160
12	2.18×10^{-20413}	3.78×10^{-681}	9.53×10^{-35}	.001462	.335672	.419706	.243160
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
20	2.18×10^{-20413}	3.78×10^{-681}	9.53×10^{-35}	.001462	.335672	.419706	.243160

Table 1: The interior rest point of the BEP($\tau^{\text{all}}, 1, \beta^{\text{min}}$) dynamic for Centipede of lengths $d \in \{3, \ldots, 20\}$. p denotes the penultimate player, q the last player. The dashed lines separate exact ($d \leq 6$) from numerical ($d \geq 7$) results.

$d=3$	$-1 \pm .3820$	-1					
$d=4$	$-1.1411 \pm .3277\,i$	$-.8589 \pm .3277\,i$					
$d=5$	$-1.1355 \pm .3284\,i$	$-.8645 \pm .3284\,i$	$-1.$				
$d=6$	$-1.1355 \pm .3284\,i$	$-.8645 \pm .3284\,i$	$-1. \pm 9.74 \times 10^{-5}\,i$				
$d=7$	$-1.1355 \pm .3284\,i$	$-.8645 \pm .3284\,i$	$-1. \pm 9.74 \times 10^{-5}\,i$	$-1.$			
$d=8$	$-1.1355 \pm .3284\,i$	$-.8645 \pm .3284\,i$	$-1. \pm 9.74 \times 10^{-5}\,i$	$-1.$	$-1.$		
$d=9$	$-1.1355 \pm .3284\,i$	$-.8645 \pm .3284\,i$	$-1. \pm 9.74 \times 10^{-5}\,i$	$-1.$	$-1.$	$-1.$	
$d=10$	$-1.1355 \pm .3284\,i$	$-.8645 \pm .3284\,i$	$-1. \pm 9.74 \times 10^{-5}\,i$	$-1.$	$-1.$	$-1.$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$
$d=20$	$-1.1355 \pm .3284\,i$	$-.8645 \pm .3284\,i$	$-1. \pm 9.74 \times 10^{-5}\,i$	$-1.$	$-1.$	$-1.$	$\cdots$

Table 2: Approximate eigenvalues of $DV(\xi^*)$ for the BEP($\tau^{\text{all}}, 1, \beta^{\min}$) dynamic. The symbol "−1." is used as a shorthand for −1.0000. The dashed lines separate exact ($d \leq 6$) from numerical ($d \geq 7$) results.

IV. Estimates of the basin of attraction of $\xi^\dagger$ for BEP($\tau^{\text{all}}, \kappa, \beta^{\min}$) dynamics in Centipede of length $d = 4$

In this section, we provide estimates of the basin of attraction of the backward induction state $\xi^\dagger$ in Centipede games of length $d = 4$ under BEP($\tau^{\text{all}}, \kappa, \beta^{\min}$) dynamics. We do so for numbers of trials ranging from $\kappa = 5$, the smallest number for which $\xi^\dagger$ is asymptotically stable (see Proposition 4.1) to $\kappa = 34$ and for selected larger values.

We estimated the size of the basin by numerically computing solutions to the BEP($\tau^{\text{all}}, \kappa, \beta^{\min}$) dynamics from points in a grid of initial conditions of mesh $\frac{1}{50}$ in the set of population states Ξ. This grid contains a total of $\binom{52}{50}^2 = 1{,}758{,}276$ points, so an exhaustive exploration is not feasible. The algorithm we used to decide which points in the grid to explore aims at "growing" the basin of attraction from $\xi^\dagger$ outwards. Specifically, we start at the vertex $\xi^\dagger$ and extend outward, recursively visiting all neighboring points in the grid until obtaining a "boundary" two-grid-points thick in which no solution converges to $\xi^\dagger$.

For $\kappa \in \{5, \ldots, 34\}$, Table 3 presents all of the grid points from which solutions of BEP($\tau^{\text{all}}, \kappa, \beta^{\min}$) dynamics converge to $\xi^\dagger$. Table 4 presents the total number of such points, as well as the sum of the number of such points and the number of neighbors of such points; these numbers provide lower and upper bounds on the size of the basin.

We make two observations about these results. First, Table 3 shows that state $\xi^\dagger$ is not at all robust to changes in the behavior of population 1. This point is reinforced in Table 5, which shows that the saddle points of the dynamics all place mass of at least .998 on strategy 1. Second, Table 4 shows that the estimated size of the basin is very small. For instance, for $\kappa = 100$, the lower and upper estimates of the size of the basin are 51 and 166 grid points, out of the total of 1,758,276 grid points.

Condition on κ	x_1	x_2	x_3	y_1	y_2	y_3
	1	0	0	1	0	0
$\kappa \geq 6$	1	0	0	0.98	0.02	0
$\kappa = 7$ or $\kappa \geq 9$	1	0	0	0.96	0.04	0
$\kappa \geq 9$	1	0	0	0.94	0.06	0
$\kappa \geq 10$	1	0	0	0.98	0	0.02
$\kappa = 10$ or $\kappa \geq 12$	1	0	0	0.96	0.02	0.02
$\kappa = 10$ or $\kappa \geq 12$	1	0	0	0.92	0.08	0
$\kappa \geq 12$	1	0	0	0.94	0.04	0.02
$\kappa = 12, 13$ or $\kappa \geq 15$	1	0	0	0.9	0.1	0
$\kappa \geq 15$	1	0	0	0.92	0.06	0.02
$\kappa = 15, 16$ or $\kappa \geq 18$	1	0	0	0.88	0.12	0
$\kappa = 15, 16, 17, 18$ or $\kappa \geq 20$	1	0	0	0.96	0	0.04
$\kappa \geq 17$	1	0	0	0.9	0.08	0.02
$\kappa = 17$ or $\kappa \geq 20$	1	0	0	0.94	0.02	0.04
$\kappa = 18, 19$ or $\kappa \geq 21$	1	0	0	0.88	0.1	0.02
$\kappa = 18$ or $\kappa \geq 21$	1	0	0	0.86	0.14	0
$\kappa \geq 20$	1	0	0	0.92	0.04	0.04
$\kappa = 20$ or $\kappa \geq 25$	1	0	0	0.94	0	0.06
$\kappa = 21$ or $\kappa \geq 24$	1	0	0	0.86	0.12	0.02
$\kappa = 22$ or $\kappa \geq 24$	1	0	0	0.9	0.06	0.04
$\kappa = 24$ or $\kappa \geq 27$	1	0	0	0.84	0.16	0
$\kappa \geq 26$	1	0	0	0.88	0.08	0.04
$\kappa = 27$ or $\kappa \geq 30$	1	0	0	0.92	0.02	0.06
$\kappa = 27$ or $\kappa \geq 30$	1	0	0	0.84	0.14	0.02
$\kappa \geq 30$	1	0	0	0.86	0.1	0.04
$\kappa = 30$ or $\kappa \geq 33$	1	0	0	0.82	0.18	0
$\kappa \geq 32$	1	0	0	0.9	0.04	0.06
$\kappa = 33$	1	0	0	0.82	0.16	0.02

Table 3: Initial conditions in a grid of mesh $\frac{1}{50}$ from which solutions of BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamics converge to $\xi^{\dagger}$ ($\kappa \in \{5, \ldots, 34\}$).

κ	# in-basin points	# in-basin points and their out-of-basin neighbors
5	1	5
6	2	9
7	3	13
8	2	9
9	4	17
10	7	27
11	5	20
12	9	34
13	9	34
14	8	30
15	12	44
16	12	44
17	13	46
18	15	54
19	13	47
20	16	56
21	18	63
22	18	63
23	17	60
24	20	70
25	20	69
26	21	72
27	24	82
28	22	76
29	22	76
30	26	89
31	25	85
32	26	88
33	28	95
34	27	92
50	35	116
100	51	166

Table 4: Number of initial conditions in a grid of mesh $\frac{1}{50}$ from which solutions of BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamics converge to $\xi^{\dagger}$, and the total number of such points and their neighbors.

V. Saddle points of BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamics in Centipede of length $d = 4$

Table 5 presents approximate components of saddle points of BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamics for Centipede games of length $d = 4$ for various κ.

	x_1	x_2	x_3
5	.999417	.000333	.000250
6	.999374	8.23×10^{-6}	.000617
7	.999093	6.76×10^{-5}	.000839
8	.999474	3.20×10^{-5}	.000494
9	.999649	1.93×10^{-7}	.000351
10	.998505	.000137	.001358
11	.998889	9.75×10^{-5}	.001013
12	.998404	4.76×10^{-5}	.001549
13	.998759	3.35×10^{-5}	.001207
14	.998926	3.65×10^{-5}	.001038
15	.998396	3.72×10^{-5}	.001567
16	.998629	3.45×10^{-5}	.001337
17	.998367	.000180	.001453
18	.998551	1.69×10^{-5}	.001432
19	.998578	3.19×10^{-5}	.001390
20	.998447	.000109	.001444
21	.998540	1.58×10^{-5}	.001444
22	.998380	6.61×10^{-5}	.001554
23	.998535	6.48×10^{-5}	.001400
24	.998484	2.03×10^{-5}	.001495
25	.998484	4.05×10^{-5}	.001476
30	.998544	1.69×10^{-5}	.001439
35	.998612	4.21×10^{-5}	.001345
40	.998669	2.03×10^{-5}	.001310
45	.998726	1.04×10^{-5}	.001264
50	.998782	2.29×10^{-5}	.001195
100	.999169	2.75×10^{-6}	.000828
150	.999368	5.23×10^{-7}	.000632
200	.999487	2.93×10^{-7}	.000513

	y_1	y_2	y_3
5	.994197	.002904	.002899
6	.992520	.003747	.003733
7	.987382	.006326	.006292
8	.991613	.004201	.004186
9	.993702	.003154	.003144
10	.970561	.014810	.014629
11	.975875	.012124	.012001
12	.962408	.018963	.018629
13	.968257	.015991	.015752
14	.970372	.014917	.014711
15	.953015	.023757	.023228
16	.957068	.021686	.021246
17	.946117	.027235	.026647
18	.949167	.025733	.025100
19	.947422	.026621	.025958
20	.939865	.030463	.029672
21	.940517	.030176	.029308
22	.931278	.034908	.033814
23	.934926	.033024	.032050
24	.929859	.035671	.034470
25	.927065	.037100	.035835
30	.916397	.042658	.040945
35	.907601	.047209	.045190
40	.899161	.051657	.049182
45	.891781	.055554	.052664
50	.885579	.058794	.055627
100	.847323	.079244	.073433
150	.827851	.089787	.082362
200	.815327	.096613	.088061

Table 5: Saddle points of BEP($\tau^{\text{all}}, \kappa, \beta^{\text{min}}$) dynamics for Centipede of length $d = 4$.

References

Akritas, A. G. (2010). Vincent's theorem of 1836: Overview and future research. *Journal of Mathematical Sciences*, 168:309–325.

Akritas, A. G., Bocharov, A., and Strzeboński, A. W. (1994). Implementation of real root isolation algorithms in *Mathematica*. In *Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering* (*Interval '94*), pages 23–27, St. Petersburg.

Alefeld, G. and Herzberger, J. (1983). *Introduction to Interval Computations*. Academic Press, New York.

Buchberger, B. (1965). *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck. Translated by M.P. Abramson as "An algorithm for finding the basis elements of the residue class ring of a zero-dimensional polynomial ideal" in *Journal of Symbolic Computation* 41 (2006), 475–511.

Buchberger, B. (1970). Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes mathematicae*, pages 374–383. Translated by M. P. Abramson and R. Lumbert as "An algorithmic criterion for the solvability of algebraic systems of equations" in B. Buchberger and F. Winkler, editors, *Gröbner Bases and Applications*, p. 535–545, 1998, Cambridge University Press.

Collins, G. E. (1975). Quantifier elimination for the theory of real closed fields by cylindrical algebraic decomposition. In *Second GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Springer, Berlin.

Collins, G. E. and Krandick, W. (1992). An efficient algorithm for infallible polynomial complex root isolation. In Wang, P. S., editor, *Proceedings of the International Symposium on Symbolic and Algebraic Computation* (*ISSAC '92*), pages 189–194, Berkeley.

Cox, D., Little, J., and O'Shea, D. (2015). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer International, Cham, Switzerland, fourth edition.

Jenkins, M. A. (1969). *Three-stage variable-shift iterations for the solution of polynomial equations with a posteriori error bounds for the zeros.* PhD thesis, Stanford University.

Jenkins, M. A. and Traub, J. F. (1970a). A three-stage algorithm for real polynomials using quadratic iteration. *SIAM Journal on Numerical Analysis*, 7:545–566.

Jenkins, M. A. and Traub, J. F. (1970b). A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration. *Numerische Mathematik*, 14:252–263.

Strzeboński, A. W. (1996). Algebraic numbers in *Mathematica 3.0*. *Mathematica Journal*, 6:74–80.

Strzeboński, A. W. (1997). Computing in the field of complex algebraic numbers. *Journal of Symbolic Computation*, 24:647–656.

Tucker, W. (2011). *Validated Numerics: A Short Introduction to Rigorous Computations*. Princeton University Press, Princeton.