Studying Continuous Time Dynamics of 2 x2 Normal – Form Games Using *Mathematica*

In this notebook, I give some examples of how to use the functions defined in my **`Continuous2x2Dynamics`** package contained in the file `Cont2x2.m`. I have used the 'Matching Pennies' game throughout this notebook, and I have evaluated the sample cells. I have included the outputs I got when I ran these examples. If you need help with the syntax of my functions, you can get online help for them by evaluating the expression (*e.g.*) **?DarwinFlowOrbit** once you have loaded the package (*i.e.*, once you have evaluated the first cell in this notebook -- see below).

The two dynamical models (*i.e.*, replicator and darwin) seen here are taken from Brian Skyrms's book *The Dynamics of Rational Deliberation*. It should be easy enough to add new dynamical flows of your own to my package. Skyrms also discusses the replicator dynamics in his more recent book *The Evolution of the Social Contract*.

First, load in the **`Continuous2x2Dynamics`** package, by evaluating the following cell (here, the path should be set appropriately to find the package fille `Cont2x2.m`).

```
<< "cont2x2.m";
Off[Solve::"svars"];
```

You can use the **?** function to obtain information about the various functions defined in the package.

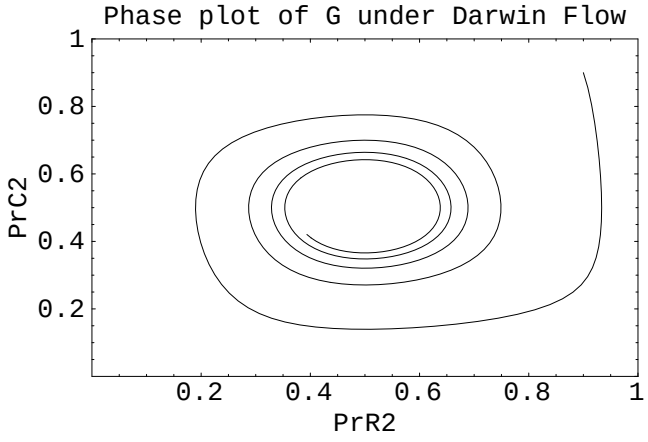
```
In[3]:= ?DarwinFlowOrbit

DarwinFlowOrbit[{G_, T_, A_, PrR20_, PrC20_}].  Takes the five-tuple <G, T, A, PrR20, PrC20>
  into a  graphics object of the orbit starting from the initial point <PrR20, PrC20> under
  the Darwin flow on the time interval [0, T]. The accuracy (i.e., step size in fourth order
  Runge Kutta numerical integration algorithm) is given by A.  G is the 2x2 Normal form
  game--it is an eight-tuple of the form: G = {R11, C11, R12, C12, R21, C21, R22, C22}.
```

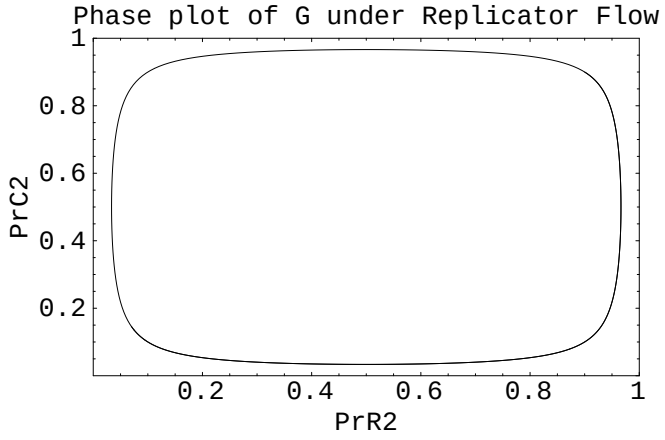
```
In[4]:= ?ReplicatorFlowOrbit

ReplicatorFlowOrbit[{G_, T_, A_, PrR20_, PrC20_}].  Takes the five-tuple <G, T, A, PrR20, PrC20>
  into a  graphics object of the orbit starting from the initial point <PrR20, PrC20> under
  the Replicator flow on the time interval [0, T]. The accuracy (i.e., step size in fourth
  order Runge Kutta numerical integration algorithm) is given by A.  G is the 2x2 Normal
  form  game--it is an eight-tuple of the form: G = {R11, C11, R12, C12, R21, C21, R22, C22}.
```

```
In[5]:= G = {R11, C11, R12, C12, R21, C21, R22, C22} = {1, 0, 0, 1, 0, 1, 1, 0};
        Print["The game in question is, G = ", TableForm[Partition[G, 4]]];

The game in question is, G =  1  0  0  1
                              0  1  1  0
```

Here, we look at an orbit of the game **G** above (*viz.*, "matching pennies") under both the darwin flow and the replicator flow.

In[7]:=
```
Sims = {{G, 30, .1, .9, .9}};
DarwinPhasePlot = DarwinFlowOrbit /@ Sims;
ReplicatorPhasePlot = ReplicatorFlowOrbit /@ Sims;
Show[{DarwinPhasePlot}, Prolog → Thickness[.0005], PlotRange → {{0, 1}, {0, 1}},
  Frame → True, FrameLabel → {"PrR2", "PrC2", "Phase plot of G under Darwin Flow", ""}];
Show[{ReplicatorPhasePlot}, Prolog → Thickness[.0005],
  PlotRange → {{0, 1}, {0, 1}}, Frame → True,
  FrameLabel → {"PrR2", "PrC2", "Phase plot of G under Replicator Flow", ""}];
```





We can also look at the vector fields set-up by the two dynamics in this game.

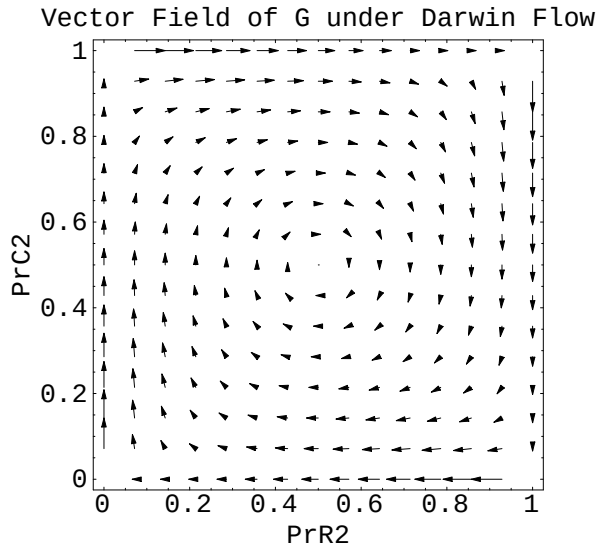
In[12]:=
```
? DarwinFlowVectorField
```

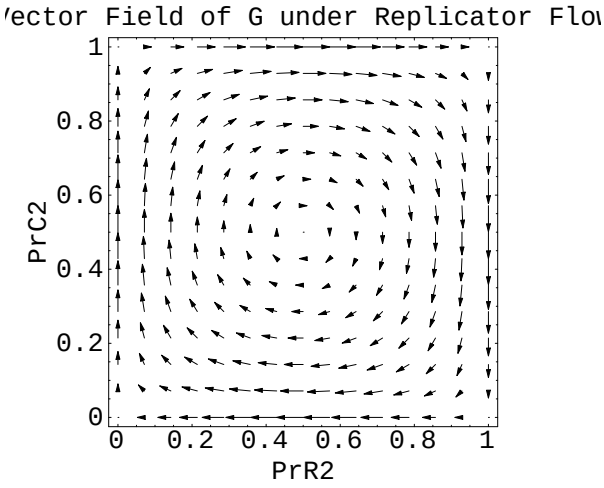
```
DarwinFlowVectorField[{G_, A_}] takes a 2x2 normal form game G and an accuracy (# of arrows
  in the vector field plot) A into a vector field plot of the Darwin flow set-up by G.
```

In[13]:=
```
? ReplicatorFlowVectorField
```

```
ReplicatorFlowVectorField[{G_, A_}] takes a 2x2
  normal form game G and an accuracy (# of arrows  in the vector field
  plot) A into a vector field plot  of the Replicator flow set-up by G.
```

In[16]:=
```
DarwinFlowVectorField[{G, 15}]
ReplicatorFlowVectorField[{G, 15}]
```





There are also functions for theoretically analysing the stability properties of equilibria in the games under the two kinds of dynamics.

In[18]:=
```
?ReplicatorFlowStability
```

```
ReplicatorFlowStability[G_], (1) finds the Nash  equilibria of a game G, and (2) performs  a
  linear stability analysis for the continuous time  Replicator flow at each of the Nash
  equilibria.   Specifically, this function outputs the real parts of  the eigenvalues
  of the Jacobian Matrix of the Replicator flow evaluated at each Nash equilibrium of G.
```

In[19]:=
```
?DarwinFlowStability
```

```
DarwinFlowStability[G_], (1) finds the Nash  equilibria of a game G, and (2) performs  a
  linear stability analysis for the continuous time  Darwin flow at each of the Nash
  equilibria.   Specifically, this function outputs the real parts of  the eigenvalues
  of the Jacobian Matrix of the Darwin flow evaluated at each Nash equilibrium of G.
```

In[20]:=
```
ReplicatorFlowStability[G]
DarwinFlowStability[G]
```

```
The Nash equilibria of G are: {{0.5, 0.5}}

The Jacobian of the Replicator Flow at the point {0.5, 0.5} is  0.    0.5
                                                               -0.5   0.

The Real Parts of the Eigenvalues of the
 Jacobian of the Replicator Flow at the point {0.5, 0.5} are: {0., 0.}

The Nash equilibria of G are: {{0.5, 0.5}}

The Jacobian of the Darwin Flow at the point {0.5, 0.5} is  0.    1.
                                                           -1.    0.

The Real Parts of the Eigenvalues of the
 Jacobian of the Darwin Flow at the point {0.5, 0.5} are: {0., 0.}
```

In this case, the standard analytical approach will not tell us whether the equilibrium point is a global attractor or not (but we do know it is *not un*stable). [As it turns out, it is a global attractor in the darwin flow case, but not in the replicator case.]

There are a couple of other useful functions in this package as well:

In[22]:=
```
G1 = {{{R11, C11}, {R12, C12}},
      {{R21, C21}, {R22, C22}}}
```

Out[22]=
```
{{{1, 0}, {0, 1}}, {{0, 1}, {1, 0}}}
```

In[23]:=
```
?IsNash
```

```
IsNash[game_,strategies_] returns True if strategies is  a Nash Equilibrium
  of game and False otherwise. Example input: IsNash[game,{{2/3,1/3},{1/3,2/3}}].
```

In[24]:=
```
IsNash[G1, {{1 / 2, 1 / 2}, {1 / 2, 1 / 2}}]
```

Out[24]=
```
True
```

In[25]:=
```
?Nash
```

```
Nash[game_] finds the Nash Equilibria of game, a game
  in  normal form. Example input: Nash[{{{2,1},{0,0}},{{0,0},{1,2}}}].
  Nash  returns the probability weights on the different pure strategies.
```

In[26]:=
```
Nash[G1]
```

Out[26]=
$$\left\{\left\{\left\{\frac{1}{2}, \frac{1}{2}\right\}, \left\{\frac{1}{2}, \frac{1}{2}\right\}\right\}\right\}$$

In[27]:=
```
Nash[{{{2, 1}, {0, 0}}, {{0, 0}, {1, 2}}}]
```

Out[27]=
$$\left\{\{\{0, 1\}, \{0, 1\}\}, \left\{\left\{\frac{2}{3}, \frac{1}{3}\right\}, \left\{\frac{1}{3}, \frac{2}{3}\right\}\right\}, \{\{1, 0\}, \{1, 0\}\}\right\}$$