Lotke-Volterra Equations: Population Models in Mathematica

Developed in the 1930s, the Lotke-Volterra Equations are a system of Ordinary Differential Equations (ODE) commonly used to model populations of species that are interacting with each other in a particular ecological environment. The most simple set of these are the Predator-Prey model, which is as follows:

$x'(t) = \alpha * x(t) - \beta * x(t) * y(t)$
$y'(t) = -\gamma * y(t) + \delta * x(t) * y(t)$

where x(t) represents the prey population as a function of time, and y(t) performs a similar role for the predator.
α is the prey growth rate, β is the effect of predator presence on prey, γ is predator death rate and δ is the effect of prey presence on predators.

In Mathematica these equations can be implemented as such:

In[ ]:=
```
(*We implement the parameters shown above first*)
a = 0.5(*prey growth rate*);
b = 1(*predator effect on prey*);
c = 1(*predator death rate*);
d = 1(*prey effect on predator*);

preyInitialPopulation = 4;
predatorInitialPopulation = 2;

duration = 100;
```

In[ ]:=
```
(*Then we can program the equations themselves*)
```

In[ ]:=
```
predatorPreyEqns = {x'[t] == a * x[t] - b * x[t] * y[t], y'[t] == -c * y[t] + d * x[t] * y[t],
  x[0] == preyInitialPopulation, y[0] == predatorInitialPopulation}
```

Out[ ]=
```
{x′[t] == 0.5 x[t] - x[t] × y[t], y′[t] == -y[t] + x[t] × y[t], x[0] == 4, y[0] == 2}
```

Unfortunately the Lotke-Volterra Equations don’t have algebraic solutions—the best we can do is numerical analysis using NDSolve[].

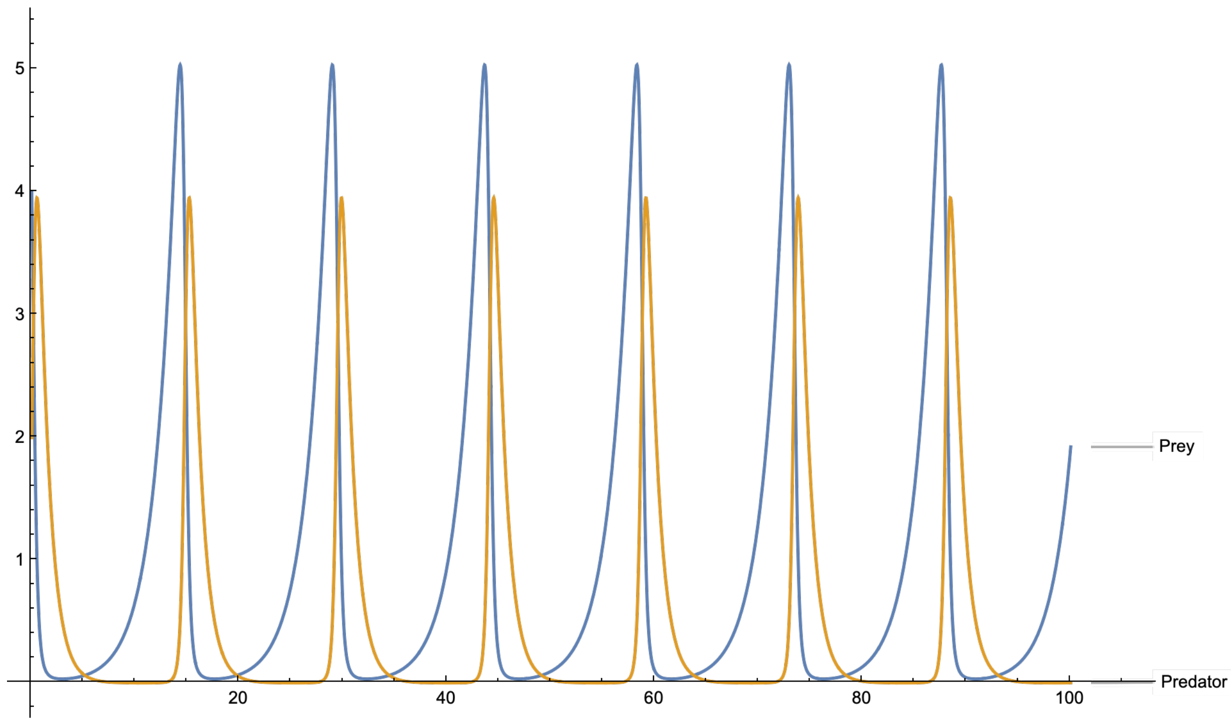
In[ ]:=
```
predatorPreySol = NDSolve[predatorPreyEqns, {x, y}, {t, 0, duration}]
```

Out[ ]=
```
{{x → InterpolatingFunction[ Domain: {{0., 100.}} Output: scalar ],
  y → InterpolatingFunction[ Domain: {{0., 100.}} Output: scalar ]}}
```

And then we can plot the solution.

In[ ]:=
```
Plot[Evaluate[{x[t], y[t]} /. predatorPreySol],
 {t, 0, duration}, PlotRange → All, PlotLabels → {"Prey", "Predator"}]
```

Out[ ]=



The next level of complexity arises when we introduce the concept of carrying capacity—which refers to the maximum population a species can take on in the environment before overpopulation begins to decrease the number of individuals. The system (known as the Lotke-Volterra Competitive Equations) becomes this:

$$x'(t) = r_1 * x(t) * \left(1 - \frac{x[t] + \alpha_{12}\, y[t]}{k_1}\right)$$
$$y'(t) = r_2 * y(t) * \left(1 - \frac{y[t] + \alpha_{21}\, x[t]}{k_2}\right)$$

where r_i and k_i refer to growth and carrying capacity respectively, and $\alpha_{i,j}$ is now redefined as the effect species j has on species i (in this case, x is 1 and y is 2).

In[ ]:=
```
(*Parameter implementation*)
r1 = 0.2;
k1 = 100;
r2 = 0.5;
k2 = 100;
α12 = 0.1;
α21 = 0.1;
xInit = 4;
yInit = 2;
```

In[ ]:=
```
compEqns = {x'[t] == r1 * x[t] * (1 - (x[t] + α12 * y[t]) / k1),
  y'[t] == r2 * y[t] * (1 - (y[t] + α21 * x[t]) / k2), x[0] == xInit, y[0] == yInit}
```

Out[ ]=

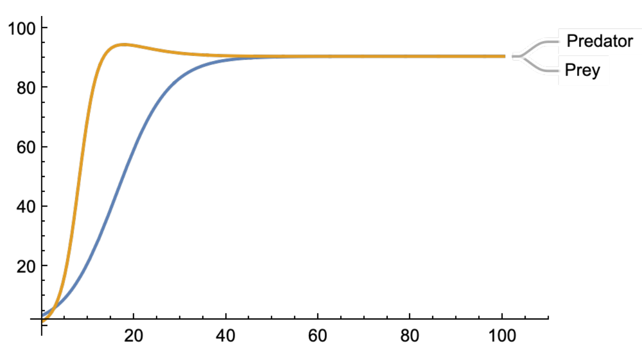
$$\left\{x'[t] == 0.2\,x[t]\left(1+\frac{1}{100}(-x[t]-0.1\,y[t])\right),\; y'[t] == 0.5\left(1+\frac{1}{100}(-0.1\,x[t]-y[t])\right)y[t],\; x[0] == 4,\; y[0] == 2\right\}$$

In[ ]:=
```
compSol = NDSolve[compEqns, {x, y}, {t, 0, duration}]
```

Out[ ]=
```
{{x → InterpolatingFunction[ Domain: {{0., 100.}} Output: scalar ],
  y → InterpolatingFunction[ Domain: {{0., 100.}} Output: scalar ]}}
```

In[ ]:=
```
Plot[Evaluate[{x[t], y[t]} /. compSol], {t, 0, duration},
 PlotRange → All, PlotLabels → {"Prey", "Predator"}]
```

Out[ ]=



The ultimate system utilised are the Generalised Lotke-Volterra Equations, which can model n number of species. This is done using the generalised formula:

$x_i'(t) = (r_i + A)\,x$

here A represents a matrix of interactions between various species. Each row of the matrix has 3 inputs, {effect of species a on species b, effect of species b on species a, and effect of species a on itself}. An example with three species is implemented below.

```
identities = {x, y, z};
growthRates = {3, 4, 7.2};
initials = {0.1, 0.8, 0.3};
interactionMatrix = {{-0.5, -1, 0}, {0, -1, -2}, {-2.6, -1.6, -3}} ;
(*Creating the interaction matrix*)
Grid[interactionMatrix] (*Visualisation*)
```

Out[∘]=

```
-0.5  -1   0
 0    -1  -2
-2.6 -1.6 -3
```

```
n = 3;
system = Table[Flatten[{identities⟦i⟧ '[t] == (growthRates⟦i⟧ +
          Total[Table[interactionMatrix⟦i⟧⟦j⟧ * identities⟦j⟧[t], {j, n}]]) *
        identities⟦i⟧[t], identities⟦i⟧ '[0] == initials⟦i⟧}], {i, n}];
(*Generating a system of equations from the parameters*)
Column[system]
```

Out[∘]=

```
{x′[t] == x[t] (3 - 0.5 x[t] - y[t]), x′[0] == 0.1}
{y′[t] == y[t] (4 - y[t] - 2 z[t]), y′[0] == 0.8}
{z′[t] == (7.2 - 2.6 x[t] - 1.6 y[t] - 3 z[t]) z[t], z′[0] == 0.3}
```

In[∘]:=

```
genEq = Flatten[system]
```

Out[∘]=

```
{x′[t] == x[t] (3 - 0.5 x[t] - y[t]), x′[0] == 0.1, y′[t] == y[t] (4 - y[t] - 2 z[t]),
 y′[0] == 0.8, z′[t] == (7.2 - 2.6 x[t] - 1.6 y[t] - 3 z[t]) z[t], z′[0] == 0.3}
```

In[∘]:=

```
genSol = NDSolve[genEq, identities, {t, 0, duration}]
```

Out[∘]=

```
{{x → InterpolatingFunction[ Domain: {{0., 100.}} Output: scalar ],
  y → InterpolatingFunction[ Domain: {{0., 100.}} Output: scalar ],
  z → InterpolatingFunction[ Domain: {{0., 100.}} Output: scalar ]}}
```

In[ ]:=
```
Plot[Evaluate[{x[t], y[t], z[t]} /. genSol],
  {t, 0, duration}, PlotLabels → {"x", "y", "z"}]
```

Out[ ]=