■ Photosynthetic Sine Wave

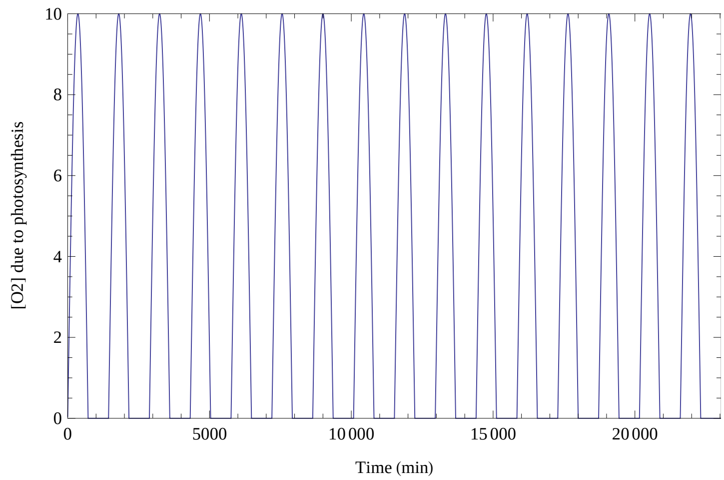
In[2]:=
```
sine[A_, F_, PSI_, time_, D_] := Module[{recov, trunk},
   recov = A * N[Sin[(2 * Pi * F * time + PSI)]] + D];
```

##############Test of sine wave over 16 days (23,040 min)##########################

In[3]:=
```
gt = Map[sine[10, .0006944444444444445, 0, #, 0] &, Table[i, {i, 1, 23 040}]];
gt2 = Map[If[# ≤ 0, 0, #] &, gt];
```

In[589]:=
```
ListLinePlot[gt2, PlotRange → {{0, 23 040}, {0, 10}}, Frame → True,
 FrameLabel -> {"Time (min)", "[O2] due to photosynthesis"}]
```

Out[589]=



■ Biological Oxygen Demand (BOD) curve

In[7]:=
```
BOD[prey_, h_] := Module[{shit},
  shit = 1 + 10 * (prey / (prey + h));
  shit]
```

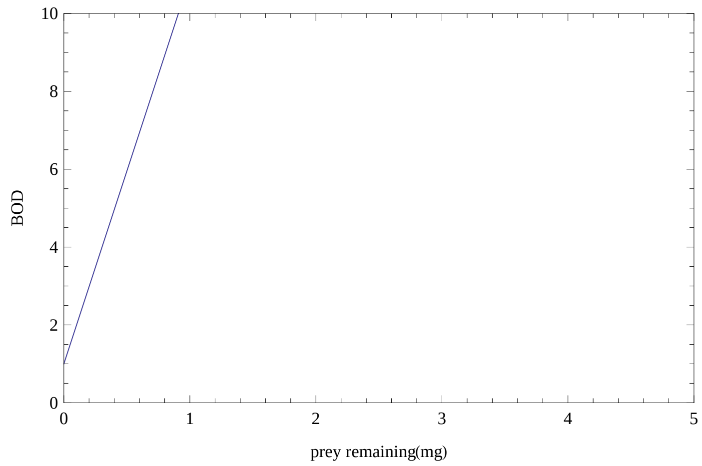
######################Test of BOD curve######################

In[420]:=
```
curve = Map[BOD[#, .01] &, Table[i, {i, 0, 5}]]
```

Out[420]=
```
{1., 10.901, 10.9502, 10.9668, 10.9751, 10.98}
```

In[422]:=
```
coupled = Partition[Riffle[ Table[i, {i, 0, 10}], curve], 2];
```

```
In[590]:= ListLinePlot[coupled, PlotRange → {{0, 5}, {0, 10}},
 Frame → True, FrameLabel -> {"prey remaining(mg)", "BOD"}]
```

Out[590]=


■ Prey Consumption Curve

```
In[16]:= pcurve[a_, b_, t_] := Module[{curve},
   curve = a * E^(-b * t)]
```

####The soulution function solves pcurve for any prey (chew) value#####

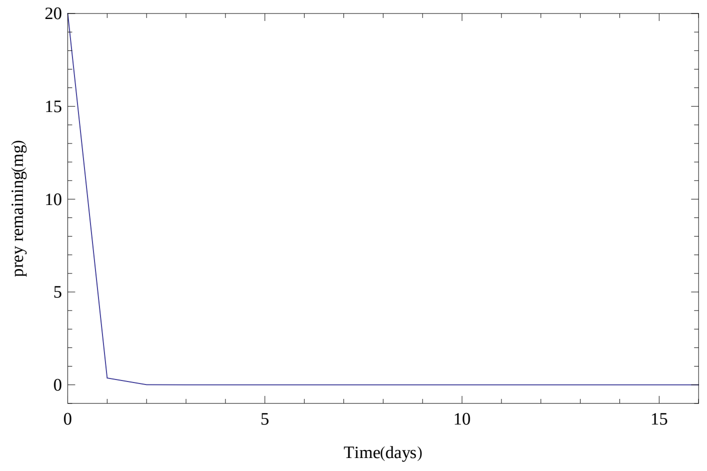
```
In[17]:= solutionFunction[chew_, day_] := Round[N[t /. Solve[pcurve[20, 4, t] == chew]] * day]

In[20]:= Round[N[t /. Solve[pcurve[20, 4, t] == 1]][[1]] * 1440]

Out[20]= 1078

In[591]:= Testpcurve =
 N[Map[pcurve[20, 4, #] &, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}]]
```

Out[591]= $\{20., 0.366313, 0.00670925, 0.000122884, 2.2507 \times 10^{-6}, 4.12231 \times 10^{-8}, 7.55027 \times 10^{-10}, 1.38288 \times 10^{-11}, 2.53283 \times 10^{-13}, 4.63905 \times 10^{-15}, 8.49671 \times 10^{-17}, 1.55623 \times 10^{-18}, 2.85033 \times 10^{-20}, 5.22056 \times 10^{-22}, 9.56179 \times 10^{-24}, 1.7513 \times 10^{-25}, 3.20762 \times 10^{-27}\}$

```
In[592]:= coup = Partition[
   Riffle[{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}, Testpcurve], 2];
```

In[595]:=
```
ListLinePlot[coup, PlotRange → {{0, 16}, {-1, 20}},
 Frame → True, FrameLabel -> {"Time(days)", "prey remaining(mg)"}]
```

Out[595]=



Nitrogen as a function of prey and oxygen

In[35]:=
```
Nitrogen[prey_, oxygen_] := Module[{N},
  N = oxygen * prey / 100]
```

################For low oxygen levels [O2]=1##############################

In[36]:=
```
one = Map[Nitrogen[#, 10] &, {0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1}]
```

Out[36]= $\left\{0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, \frac{1}{10}\right\}$

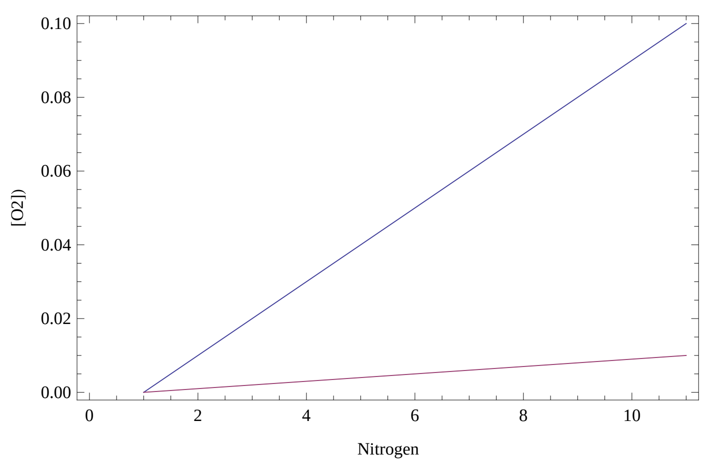
################For high oxygen levels [O2]=9##############################

In[37]:=
```
nine = Map[Nitrogen[#, 1] &, {0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1}]
```

Out[37]= $\left\{0, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, \frac{1}{100}\right\}$

In[597]:=
```
ListLinePlot[{one, nine}, Frame → True, FrameLabel -> {"Nitrogen", "[O2])"}]
```

Out[597]=



■ Oxygen augmentation to the sine curve as a function of mineralized N

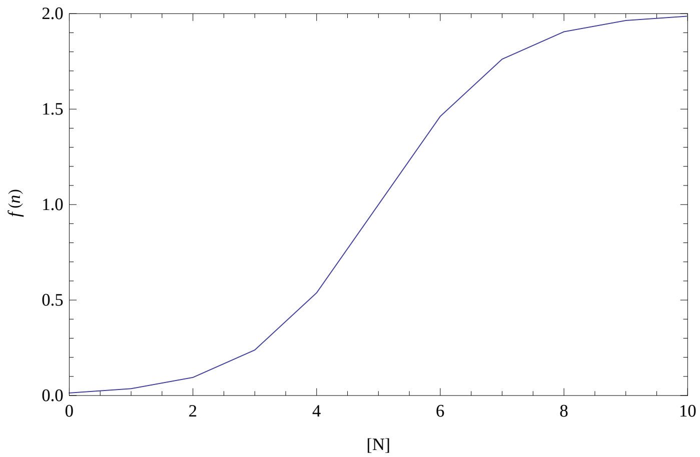
In[39]:=
```
sig[MO_, MX_, a_, x_, d_] := Module[{one, two, three},
  one = a (x - d);
  two = N[E^one];
  three = (MO) + ((MX - MO) / (1 + two));
  three]
```

In[40]:=
```
feedback = Map[sig[0, 2, -1, #, 5] &, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}];
```

In[41]:=
```
coupled = Partition[Riffle[{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, feedback], 2];
```

In[42]:=
```
ListLinePlot[coupled, PlotRange → {{0, 10}, {0, 2}},
 Frame → True, FrameLabel -> {"[N]", "f(n)"}]
```

Out[42]=



############Augmentation Function###########################

```
Aug[prey_, oxy_] := Module[{Aug, N},
  N = Nitrogen[prey, oxy];
  Aug = sig[0, 2, -1, N, 5];
  Aug]
```

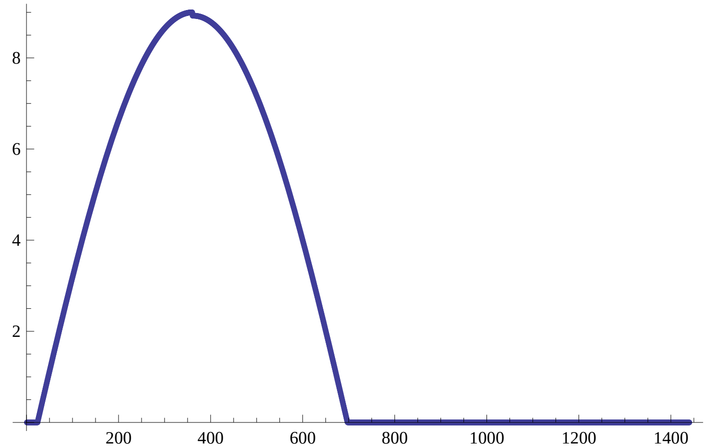
■ Dynamics of entire model for 1 day (1440 min)

In[573]:=
```
Day[{previousChow_, preyloss_, previousSpot_, prevO2_}, addchow_, lastday_, h_,
   Aug_, b_] := Module[{index, o2, newChow, sin, nullsin, bod, remove, nullo2,
    tfake, curveTime2, curveTime, t, chew, left, BigcurveTime, ppChow2},
   index = previousSpot + 1;
   sin = sine[10 * Aug, .0006944444444444445, 0, index, 0];
   nullsin = If[sin ≤ 0, 0, sin];
   ppChow2 = If[index == 360, addchow + previousChow, previousChow];
   BigcurveTime =
    If[index < 360, Round[N[t /. Solve[pcurve[20, b, t] == lastday]][[1]] * 1440],
     Round[N[t /. Solve[pcurve[20, b, t] == addchow + lastday]][[1]] * 1440]];
   tfake = If[ppChow2 == 0, BigcurveTime, BigcurveTime + index];
   remove = N[pcurve[20, b, BigcurveTime * 0.0006944444444444445]] -
     N[pcurve[20, b, tfake * 0.0006944444444444445]];
   left = If[remove == 0, ppChow2, pcurve[20, b, tfake * 0.0006944444444444445]];
   bod = BOD[left, h];
   o2 = nullsin - bod;
   nullo2 = If[o2 < 0, 0, o2];
   {left, remove, index, nullo2}]
```

In[574]:=
```
O2Dynamics = NestList[Day[#, .1, 0, 5, 1, 4] &,
    {0, 0, 0, sine[10 * 1, .0006944444444444445, 0, 0, 0]}, 1439];
```

In[575]:=
```
or = ListPlot[Map[#[[4]] &, O2Dynamics]]
```

Out[575]=



■ Looping theDynamics for all 16 days

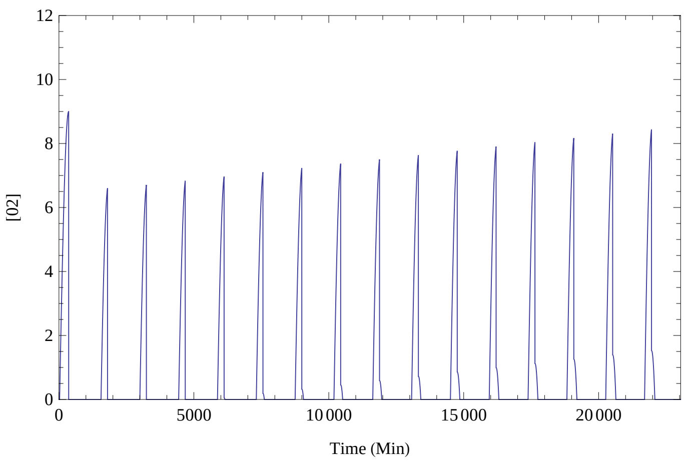
In[576]:=
```
dayPlus[{Augment_, old_, lastday_}, addchow_, b_] :=
 Module[{O2Dynamics, outs, aug, finalO2, nchow, leftoverChow},
  O2Dynamics = NestList[Day[#, addchow, lastday, .1, Augment, b] &,
    {lastday, 0, 0, sine[10 * Augment, .0006944444444444445, 0, 0, 0]}, 1439];
  outs = O2Dynamics[[All, {1, 2, 4}]];
  finalO2 = outs[[1440]][[3]];
  nchow = outs[[1440]][[2]];
  leftoverChow = outs[[1440]][[1]];
  aug = Aug[nchow, finalO2] + Augment;
  {aug, outs, leftoverChow}]
```

```
In[578]:= run = NestList[dayPlus[#, 5, 4] &, {1, 1, 0}, 16];
```

- **Process data and plot**

```
In[581]:= fr = Partition[Flatten[Map[run[[#]][[2]] &, Table[i, {i, 2, 17}]]], 3];

In[582]:= Dimensions[fr]

Out[582]= {23 040, 3}

In[583]:= wchow = Map[#[[1]] &, fr];

In[598]:= o2 = Map[#[[3]] &, fr];

In[599]:= ListLinePlot[o2, PlotRange → {{0, 23 040}, {0, 12}},
  Frame → True, FrameLabel -> {"Time (Min)", "[02]"}]
```

Out[599]=




```
In[600]:= ListLinePlot[wchow, PlotRange → {{0, 23 500}, {0, 20}},
  Frame → True, FrameLabel -> {"Time(Min)", "Prey Remaining (mg)"}]
```

Out[600]=