1 > #3 Species with Variable Parms : v parms *users more sensitive*

```
2 > restart : with(plots) : with(StringTools) :
    with(DEtools) : with(ColorTools) : with(plots) : with(geom3d) : with(plottools) :
        with(FileTools) : with(Optimization) : with(VectorCalculus) : with(LinearAlgebra) :
        with(geom3d) :
3 > FormatTime("%I:%M-%p----%d-%b-%Y"); currentdir( ); printlevel := 1 :
                                "09:59-AM----20-May-2020"
        "C:\Users\nn\Documents\2 research\2017 summer non const parms\current"                (1)
4. > ###### TOP ############
5. > #Analytic Solution 2D
6. > #Analytic Solution 3D
7. > #Parameters
8. > #s=10
9. > #Jacobian
10. > #Parameters
11. > unassign('s') :
12. > e1 := d/dt U(t) = -u1(s)·U(t) + u2(s)·V(t)·U(t)·(1 - U(t)/ku) :
13. > e2 := d/dt V(t) = -v1·V(t) + v2·V(t)·U(t)·(1 - V(t)/kv) - v3·A(t)·V(t) :
14. > e2b := d/dt V(t) = -v1·V(t) + v2·V(t)·U(t)·(1 - V(t)/kv) :
15. > e3 := d/dt A(t) = -a1(s)·A(t) + a2(s)·V(t)·A(t) :
16. > ############ 3D system ###################
17. > Udot3d(s) := simplify(rhs(e1)/U(t)) :
18. > Vdot3d(s) := simplify(rhs(e2)) :
19. > Adot3d(s) := simplify(rhs(e3/A(t))) :
20. > Adot3D(s) := subs({V(t) = V}, Adot3d(s)) :
21. > Udot3D(s) := subs({V(t) = V, U(t) = U}, Udot3d(s)) :
22. > Vdot3D(s) := subs({V(t) = V, U(t) = U}, Vdot3d(s)) :
23. > ### solve without time dependence for analytic solution ######################
24. > Vs3D := s→solve(Adot3D(s) = 0, V) :
25. > Us3D := s→simplify(subs(V(t) = Vs3D(s), solve(Udot3d(s) = 0, U(t)))) :
26. > As1 := s→solve(subs(A(t) = A, Vdot3D(s)) = 0, A) :
27. > As2 := s→subs(U = Us3D(s), As1(s)) :
28. > As3D := s→simplify(subs(V = Vs3D(s), As2(s))) :
29. > Astar := s→As3D(s) : Vstar := s→Vs3D(s) : Ustar := s→Us3D(s) :
30. > unassign('ku','kv','v1','v2','v3','u1','u2','a1','a2','s')
31. > Vdot2D := s→ -v1 + v2·U·(1 - V/kv) :;
32. > Udot2D := s→ -u1(s) + u2(s)·V(1 - U/ku) :
```

```
> ### 2D    equations ##########
> eqV2D := s→Vdot2D(s) = 0 :
> eqU2D := s→Udot2D(s) = 0 :
> UVsol := s→solve( [eqV2D(s), eqU2D(s) ], [U, V ] ) :
> ###Parms Depend on S with logistic curve FROM EVOLUTION
```

$$> a1 := s \rightarrow \frac{La1}{1 + e^{-ma1\,(s - ca1)}} :$$

$$> a2 := s \rightarrow La2 - \frac{La2}{1 + e^{-ma2\,(s - ca2)}} :$$

$$> u1 := s \rightarrow \frac{Lu1}{1 + e^{-mu1\,(s - cu1)}} :$$

$$> u2 := s \rightarrow Lu2 - \frac{Lu2}{1 + e^{-mu2\,(s - cu2)}} :$$

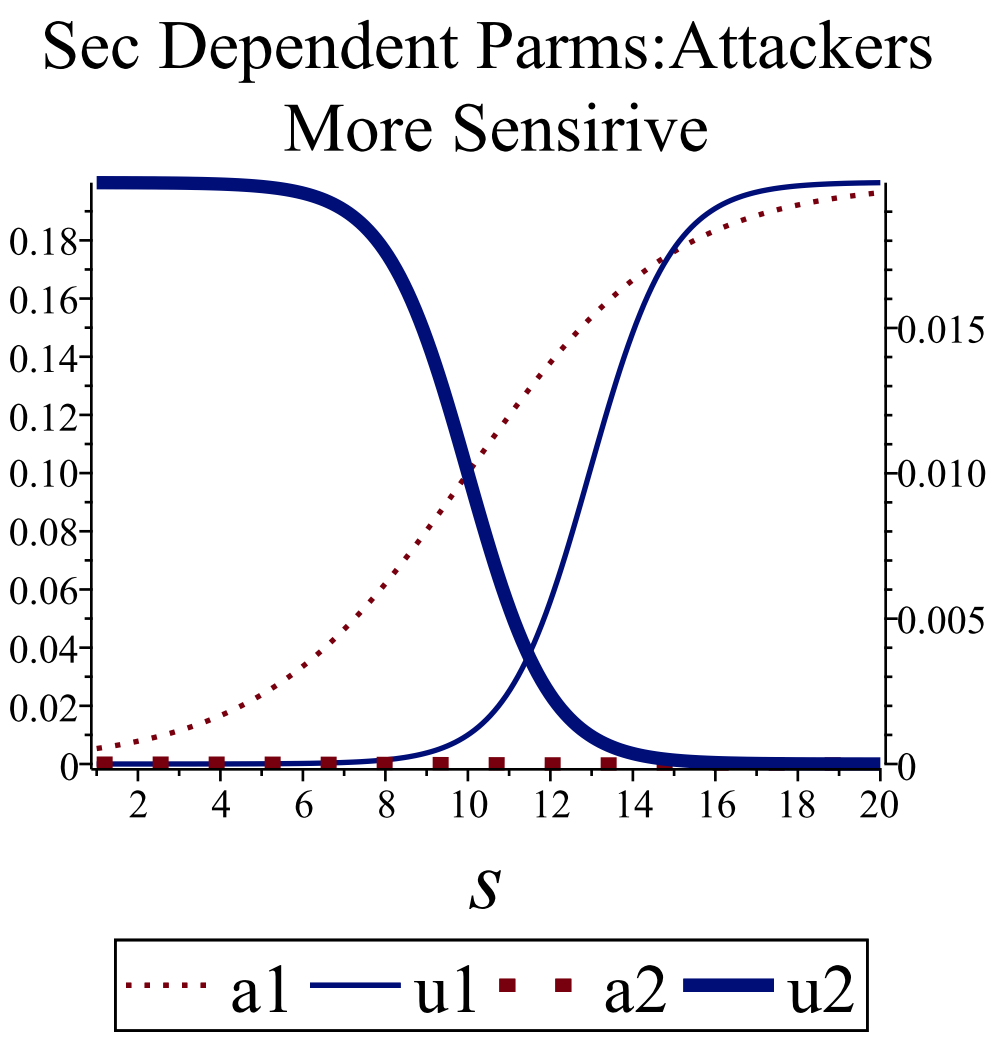
```
> ################     original parms          ####################
> ku := 10000 : kv := 30000 :  v1 := 0.001 : v2 := 0.03 : v3 := 0.1 :
> La1 := 0.2 :          ma1 := 0.4 :     ca1 := 10 :
> La2 := 0.00003 :   ma2 := 0.6 :     ca2 := 8.845 :
> Lu1 := 0.2 :      mu1 := 1 :       cu1 := 12.944 :
> Lu2 := 0.02 :      mu2 := 1 :     cu2 := 10 :
> ##########################################################
> a1(10); a2(10); evalf(u1(10)); evalf(u2(10))
> unassign('s')
> A1plot := plot( [a1(s), u1(s) ], s = 1 ..20, thickness = [2, 2 ], legend = ["a1", "u1"], title
      = "Sec Dependent Parms:Attackers More Sensirive", linestyle = [dot, solid],
      legendstyle = [ font = [roman, 20 ]], titlefont = [roman, 20 ], labelfont = [roman,
      25 ], axis[2 ] = [location = low ]) :
> A2plot := plot( [a2(s), u2(s) ], s = 1 ..20, thickness = [5, 5 ], legend = ["a2", "u2"], title
      = "Sec Dependent Parms: Attackers More Sensitive ", linestyle = [dot, solid],
      legendstyle = [ font = [roman, 20 ]], titlefont = [roman, 20 ], labelfont = [roman,
      25 ], axis[2 ] = [location = high ]) :
> dualaxisplot(A1plot, A2plot)
```

0.1000000000

0.00001000098123

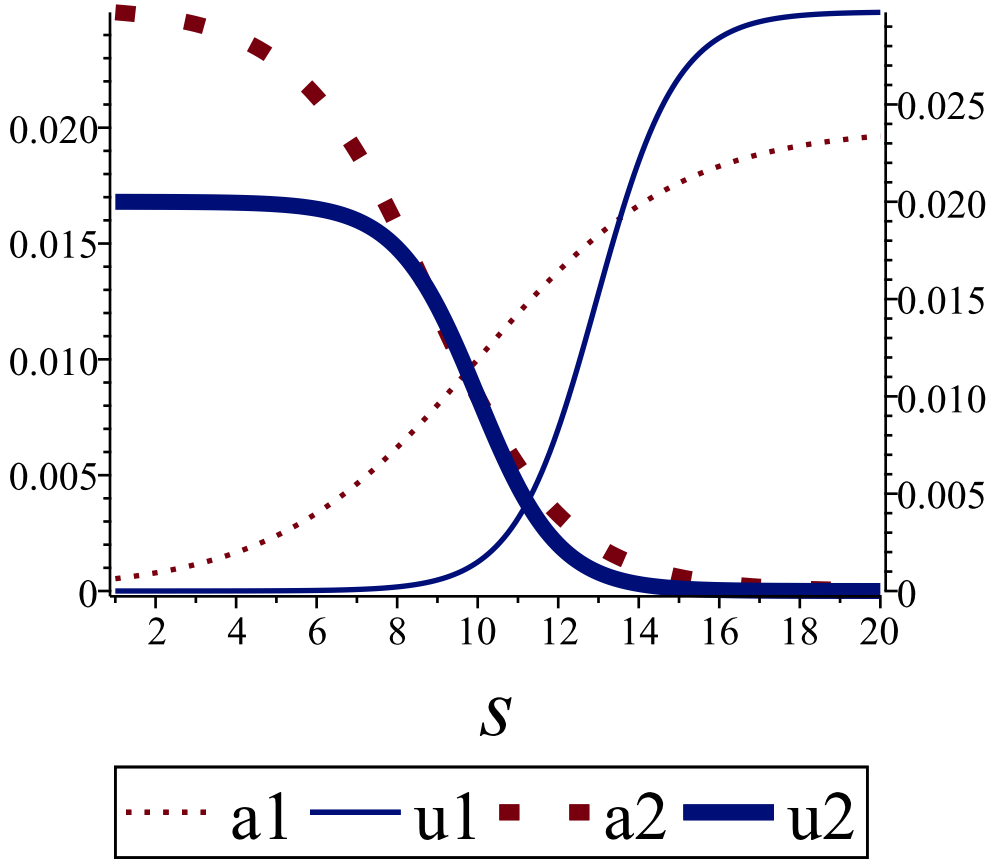
0.01000417112

0.01000000000




```
54. > ############ USER MORE SENSITIVE ###################################
55 > Lu1 := 0.025 :
56 > Lu2 := 0.02 :
57 > La1 := 0.02 :
58 > La2 := 0.03 :
59 > A1plot := plot( [a1(s), u1(s) ], s = 1 ..20, thickness = [2, 2 ], legend = ["a1", "u1"], title
          = "Sec Dependent Parms: Users More Sensitive", linestyle = [dot, solid], legendstyle
          = [ font = [roman, 20 ]], titlefont = [roman, 20 ], labelfont = [roman, 25 ], axis[2 ]
          = [location = low]) :
60 > A2plot := plot( [a2(s), u2(s) ], s = 1 ..20, thickness = [6, 6 ], legend = ["a2", "u2"], title
          = "Sec Dependent Parms: Users More Sensitive", linestyle = [dot, solid], legendstyle
          = [ font = [roman, 20 ]], titlefont = [roman, 20 ], labelfont = [roman, 25 ], axis[2 ]
          = [location = high]) :
61 > dualaxisplot(A1plot, A2plot)
62. > ##########################################################################\
          #############
```

Sec Dependent Parms: Users More Sensitive




```
63. > ###############BASINS OF ATTACTION USER MORE SENSITIVE
         ##########################
64. > L := [1, 2, 7, 10, 12]; Outlist := [ ] : print ("s", "origin", "A,U,V 2 D soln",
         "A,U,V 3D solution") :
```

$$L := [1, 2, 7, 10, 12] \quad \textbf{(2)}$$

```
65. > for s in L do
66. > ########################LOOP on Security
67. > dmax := 10 :
      Tend := 150 :
         ## difference for ending,  max time  # tend different from earlier.  For L=10
         singularity at ~200
68. > Vmax := 30000 : Umax := 12000 : Amax := 1500 : Vstep := 5000 : Ustep := 3000 :
      Astep := 500 :
69. > point(origin, [0, 0, 0]) :
70. > Ustar2 := evalf(rhs(UVsol(s)[2, 1])) :
71. > Vstar2 := evalf(rhs(UVsol(s)[2, 2])) :
72. > point(Spoint3, As3D(s), Us3D(s), Vs3D(s)) : Astar3 := evalf(As3D(s)) :
73. > point(Spoint2, 0, Ustar2, Vstar2) :
74. > C3 := coordinates(Spoint3); C2 := coordinates(Spoint2);
      C0 := coordinates(origin);
```

75. > ### Create output list of CP ##############################

```
76. > print(s, C0, C2, C3);
77. > #Outlist:=[op(Outlist) + [s, C0, C2, C3]]
78. > Vdelta := (Vmax/Vstep + 1) : Adelta := (Amax/Astep + 1) :
```

79. > ####### need to ensure that Spoints have the same ordering as

```
80. > for Ustart from 0 by Ustep to Umax do            ## loop on starting points
81. > for Astart  from 0 by Astep to Amax do
82. > for Vstart  from 0 by Vstep to Vmax do
83. >       i := Vstart/Vstep + Astart/Astep·Vdelta + Ustart/Ustep·Adelta·Vdelta :
84. > if Vstart·Ustart = 0 then Senda := 0; Sendv := 0; Sendu := 0;
          ## for starting point, calculate solution
85. > print ("line  74")

86. > else
87. >       soln := dsolve({e1, e2, e3, V(0) = Vstart, U(0) = Ustart, A(0) = Astart},
          maxfun = 0, numeric) :
88. >       Send := soln(Tend) :
89. >       Senda := rhs(Send[2]) : Sendu := rhs(Send[3]) : Sendv := rhs(Send[4]) :
90. > end if;
91. > ###print(Senda, Sendu, Sendv)
          ## select symbol for solution at the starting point
92. >       point(Spoint, Senda, Sendu, Sendv) :
93. >       color := white;
94. >       d0 := distance(origin, Spoint);
95. >       d2 := distance(Spoint2, Spoint) :
96. >       d3 := distance(Spoint3, Spoint) :
97. >       if d2 ≤ dmax then c :='green' end if:
98. >       if d3 ≤ dmax then c :='red'  end if:
99. >       if d0 ≤ dmax then c :='black'  end if:
100. >        ColorArray[i] := (Astart, Ustart, Vstart, c, d0, d3, d2) :
101. > end do; end do; end do;
```

102. > ########### ***print assignments*****######**
########################

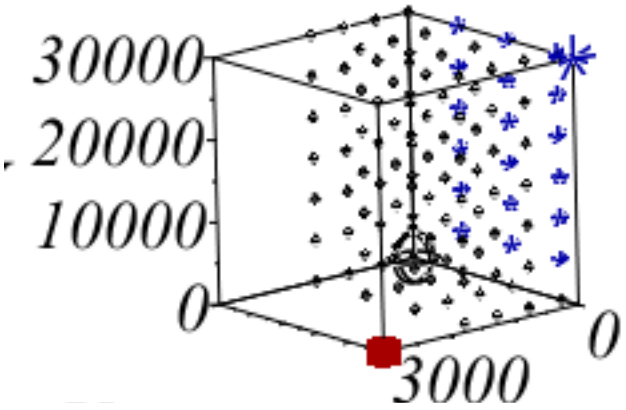
```
103. > LB := [ ] : LR := [ ] : LG := [ ] : cb := 0 : cr := 0 : cg := 0 : s3 := s
104. > for Ustart from 0 by Ustep to Umax do          ##loop thru starting points
105. > for Astart  from 0  by Astep to Amax  do
106. > for Vstart  from 0 by Vstep to Vmax do
107. >  i := Vstart/Vstep + Astart/Astep·Vdelta + Ustart/Ustep·Adelta·Vdelta :
```

108. > ############### **create array for the symbols**

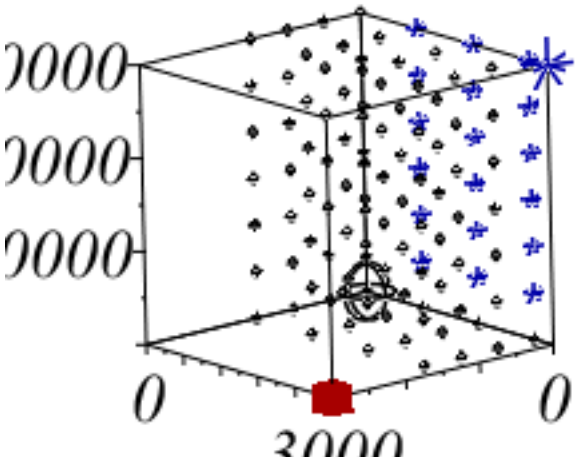
```
109. > if ColorArray[i][4]='black'  then cb := cb + 1 :  LB := [op(LB),
          [(ColorArray[i][1], ColorArray[i][2], ColorArray[i][3])]]; end if;
110. > if ColorArray[i][4]='green'  then cg := cg + 1 :  LG := [op(LG),
          [(ColorArray[i][1], ColorArray[i][2], ColorArray[i][3])]]; end if;
111. > if ColorArray[i][4]='red'    then cg := cg + 1 :  LR := [op(LR),
```

```
            [(ColorArray[i][1], ColorArray[i][2], ColorArray[i][3])]]; end if;
112. > end do;end do;end do;
113. > unassign('color') :        ########## create plots
114. > p0 := pointplot3d([(0, 0, 0)], view = [0..10, 0..10, 0..10], color = black,
            symbolsize = 100, symbol = circle) :
115. > p3 := pointplot3d([coordinates(Spoint3)], labels = ["A", "U", "V"], view = [0
            ..12000, 0..10000, 0..30000], color = "DarkRed", symbolsize = 50, symbol
            = solidbox) :
116. > p2 := pointplot3d([coordinates(Spoint2)], labels = ["A", "U", "V"], color
            = "DarkBlue", symbolsize = 200, symbol = asterisk) :
117. > pR := pointplot3d(LR, color = red, symbol = box, symbolsize = 20) :
118. > pB := pointplot3d(LB, color = [black], symbol = circle, symbolsize = 15) :
119. > pG := pointplot3d(LG, color = "DarkBlue", symbol = asterisk, symbolsize = 80) :
120. > graph||s := display(pR, pG, pB, p3, p2, p0, labels = ['A','U','V'], orientation
            = [45, 75, -5], labelfont = [Roman, italic, 20], font = [Roman, italic, 20], title
            = [typeset(cat("s=", s), cat("    dmax = ", dmax))], view = [0..3000, 0
            ..10000, 0..30000])
121. > end do:
  1, [0, 0, 0], [0, 9999.999997, 29999.90000], [2998.626071, 9995.459529, 0.01789147545]
  2, [0, 0, 0], [0, 9999.999993, 29999.90000], [2997.490762, 9991.678046, 0.02654018394]
  7, [0, 0, 0], [0, 9999.998856, 29999.90000], [2949.838978, 9832.897226, 0.2053260975]
  10, [0, 0, 0], [0, 9999.958316, 29999.90000], [2624.709287, 8749.355905, 0.9999018866]
  12, [0, 0, 0], [0, 9999.020948, 29999.89999], [492.3577137, 1641.417973, 3.513928063]   (3)
122. > for s in L do graph||s end do;
```

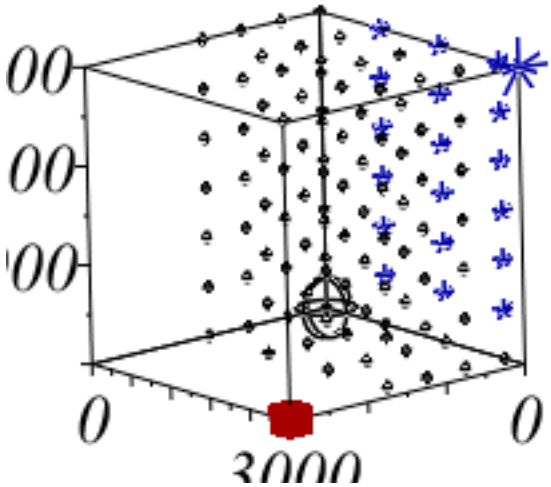
$$s=1 \quad dmax = 10$$



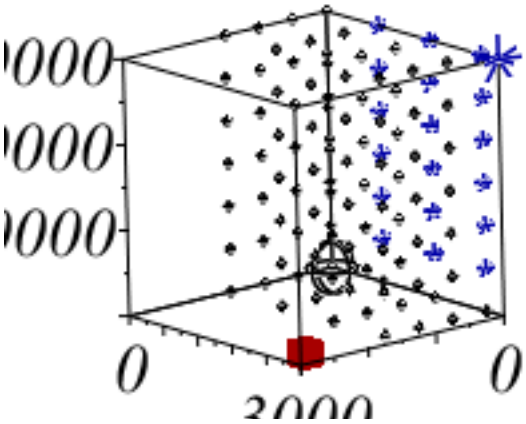
$s=2 \quad dmax = 10$



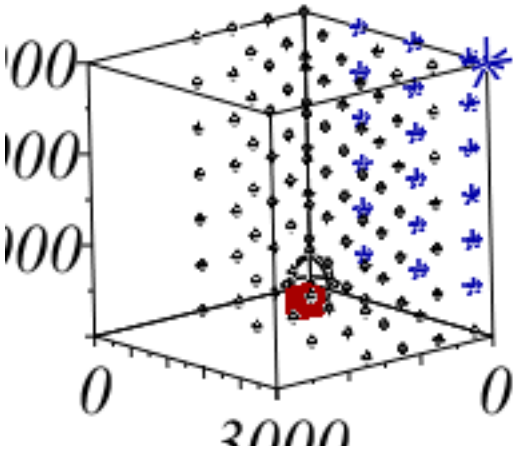
$s=7 \quad dmax = 10$



$s=10 \quad dmax = 10$



$$s=12 \quad dmax = 10$$




```
123. > save graph1, graph2, graph7, graph10, graph12, "graphs.mw"
```