

Projekt 2

Marcin Skrzypczak
Modelowanie Matematyczne
Prowadzący: dr inż. Jakub Wagner

Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych
Warszawa
30.01.2022

Oświadczam, że niniejsza praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu Modelowanie matematyczne, została wykonana przeze mnie samodzielnie.
Marcin Skrzypczak, 320735

Spis treści

1	Sformułowanie zadania	2
2	Modelowanie populacji ofiar	2
3	Modelowanie populacji drapieżników	3
4	Dopasowanie modelu	4
5	Stan równowagi	4
6	Modelowanie populacji Chromistów	5
7	Wykorzystane programy	6

1 Sformułowanie zadania

Równania Lotki-Volterry służą do modelowania liczebności populacji dwóch gatunków, między którymi występuje zależność drapieżnik-ofiara:

$$\frac{dx}{dt} = r_x x(t) + r_{xy} x(t)y(t) + r_{xx} x^2(t) \quad (1)$$

$$\frac{dy}{dt} = r_y y(t) + r_{yx} x(t)y(t) + r_{yy} y^2(t) \quad (2)$$

gdzie x to liczebność populacji gatunku ofiary, y - liczebność populacji gatunku drapieżnika, t - czas, $r_x, r_y, r_{xy}, r_{yx}, r_{xx}, r_{yy} \in R$ - parametry modelu, t_1, \dots, t_N - chwile, w których dokonano pomiaru oraz $\tilde{x}_1, \dots, \tilde{x}_N$ i $\tilde{y}_1, \dots, \tilde{y}_N$ wyniki pomiaru. Zadanie skupia się na numerycznym przybliżeniu parametrów modelu opisującego zadane dane. Obliczenia zostały przeprowadzone przy użyciu następujących metod rozwiązywania RRZ:

1. jawnej metody Eulera:

$$\hat{x}_n = \hat{x}_{n-1} + f(t_{n-1}, \hat{x}_{n-1})\Delta t$$

2. jawnej metody Adamsa-Bashfortha trzeciego rzędu:

$$\hat{x}_n = \hat{x}_{n-1} + \frac{1}{12} [23f(t_{n-1}, \hat{x}_{n-1}) - 16f(t_{n-2}, \hat{x}_{n-2}) + 5f(t_{n-3}, \hat{x}_{n-3})] \Delta t$$

3. niejawnej metody Eulera

$$\hat{x}_n = \hat{x}_{n-1} + f(t_n, \hat{x}_n)\Delta t$$

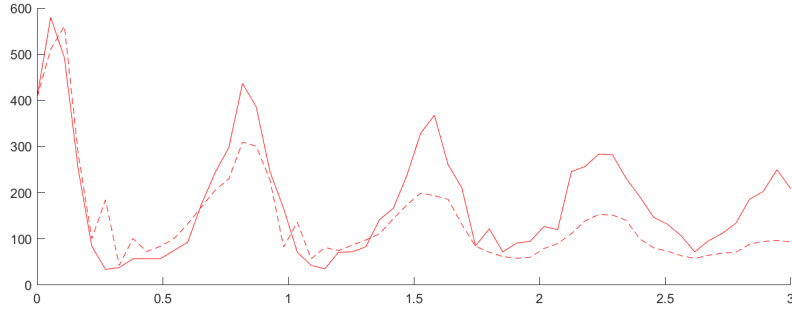
2 Modelowanie populacji ofiar

Początkowym celem będzie wyznaczenie wartości r_x, r_{xy} i r_{xx} minimalizujących wskaźnik dopasowania modelu do danych, określonego wzorem

$$J_x \equiv \sum_{n=2}^N (\hat{x}_n - \tilde{x}_n)^2,$$

gdzie \hat{x}_n ($n = 2, \dots, N$) oznacza estymatę wartości $x(t_n)$, uzyskaną poprzez rozwiązanie równania (1) po podstawieniu $y(t_1) = \tilde{y}_1, \dots, y(t_N) = (\tilde{y})_N$ oraz $x(t_1) = \tilde{x}_1$.

Najdokładniejsze przybliżenie zostało otrzymane przy użyciu metody Adamsa-Bashfortha. Parametrami opisującymi dobrany model są $r_x = 6.14$, $r_{xy} = -0.07$, $r_{xx} = 0.00$.



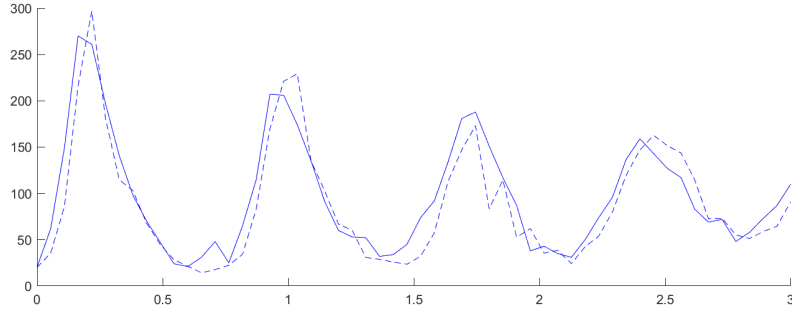
Rysunek 1: Rozmiar populacji ofiar w zależności od czasu. Linia ciągłą przedstawiono pomiary rzeczywiste, przerywaną estymatę.

3 Modelowanie populacji drapieżników

Następnie wyznaczono wartości r_y , r_{yx} i r_{yy} minimalizujące wskaźnik dopasowania modelu do danych, określonego wzorem

$$J_y \equiv \sum_{n=2}^N (\hat{y}_n - \tilde{y}_n)^2,$$

gdzie \hat{y}_n ($n = 2, \dots, N$) oznacza estymatę wartości $y(t_n)$, uzyskaną poprzez rozwiązanie równania (2) po podstawieniu $x(t_1) = \tilde{x}_1, \dots, x(t_N) = \tilde{x}_N$ oraz $y(t_1) = \tilde{y}_1$. Najdokładniejsze przybliżenie zostało otrzymane przy użyciu metody Adamsa-Bashfortha. Parametrami opisującymi dobrany model są $r_y = -12.79$, $r_{yx} = 0.07$, $r_{yy} = 0.02$.



Rysunek 2: Rozmiar populacji drapieżników w zależności od czasu. Linia ciągłą przedstawiono pomiary rzeczywiste, przerywaną estymatę.

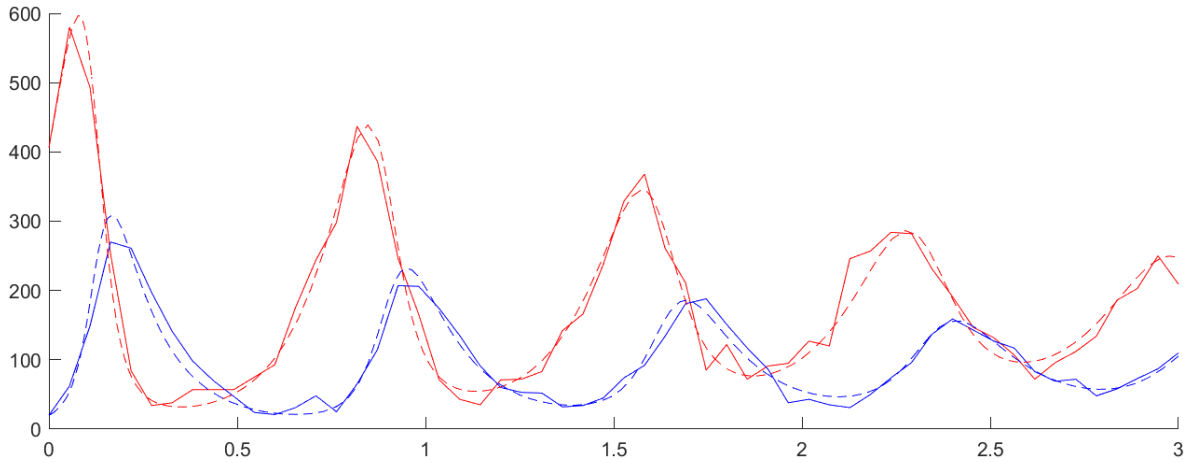
4 Dopasowanie modelu

Poprzednio wyznaczone wartości parametry posłużyły do dopasowania modelu, minimalizującego wskaźnik dopasowania, określony wzorem

$$J \equiv \sum_{n=2}^N (\hat{x}_n - \tilde{x}_n)^2 + \sum_{n=2}^N (\hat{y}_n - \tilde{y}_n)^2,$$

gdzie $\hat{x}_n, \hat{y}_n (n = 2, \dots, N)$ oznaczają estymat wartości $x(t_n), y(t_n)$, uzyskane poprzez rozwiązanie równań (1), (2) dla warunków początkowych $x(t_1) = \tilde{x}_1, y(t_1) = \tilde{y}_1$. Jako wartości początkowe zostały użyte wartości wyznaczone przy pomocy metody Eulera, ponieważ zapewniały one dokładniejszy wynik końcowy ($r_x = 6.136346, r_{xy} = -0.069893, r_{xx} = 0; r_y = -5.884900, r_{yx} = 0.059108, r_{yy} = -0.033141$). Do wyznaczenia wartości parametrów została użyta funkcja **fminsearch**, a do rozwiązywania układu równań funkcja **ode45**. Następujące parametry określają najlepszy model:

r_x	r_{xy}	r_{xx}	r_y	r_{yx}	r_{yy}
9.2205	-0.0977	0.0001	-8.1276	0.0545	-0.0098



Rysunek 3: Rozmiar populacji drapieżników i ofiar w zależności od czasu. Linia ciągłą przedstawiono pomiary rzeczywiste, przerywaną estymatę. Kolorem czerwonym oznaczono populację ofiar, niebieskim drapieżników.

5 Stan równowagi

Dla wyznaczonych wartości parametrów możliwe jest wyznaczenie wartości $x(t), y(t)$ dla których układ osiąga stan równowagi, są one rozwiązaniem układu równań (3).

$$\begin{aligned} 0 &= r_x x + r_{xy} xy + r_{xx} x^2 \\ 0 &= r_y y + r_{yx} xy + r_{yy} y^2 \end{aligned} \tag{3}$$

Którego rozwiązaniem jest para $x = 166.1312, y = 94.5457$.

6 Modelowanie populacji Chromistów

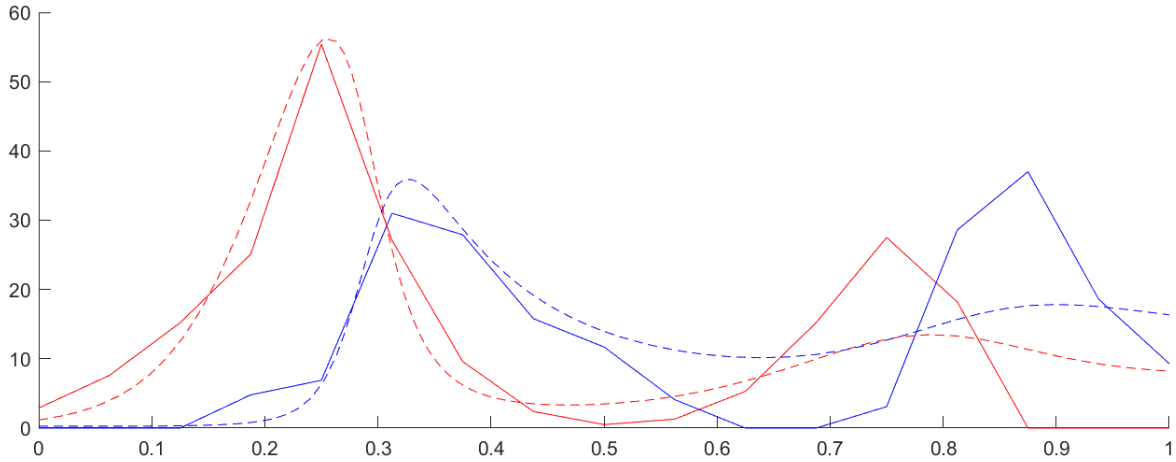
Parametry r_x i r_y opisują naturalne tempo wzrostu populacji w optymalnych warunkach, skutkujące eksponencjalnym przyrostem. Uwzględnienie r_{xx} oraz r_{yy} pozwala wyznaczyć maksymalny rozmiar populacji, spowodowany przykładowo pełnym wykorzystaniem zasobów.

$$\frac{dx}{dt} = r_x x(t) \implies x(t) = x_1 e^{r_x t}, \quad \frac{dx}{dt} = r_x x(t) + r_{xx} x^2(t) \implies x(t) = x_1 \frac{1 - r_{xx}}{r_x} \frac{r_x e^{r_x t}}{1 - r_{xx} e^{r_x t}}$$

Parametry r_{xy}, r_{yx} opisują relację międzygatunkową. Spodziewany jest negatywny wpływ drapieżników na populację ofiar ($r_{xy} < 0$) oraz pozytywny ofiar na populację drapieżników ($r_{yx} > 0$).

W zadaniu zostały wykorzystane dane reprezentujące pomiary liczebności populacji chromistów *Paramecium Caudatum* i *Didinium Nasutum* w pewnym doświadczeniu z 1934 r. Dodatkowo wartości początkowe \hat{x}_1 i \hat{y}_1 zostały dobrane, aby minimalizować błąd estymacji. Aby zwiększyć dokładność modelowania dane zostały przeskalowane liniowo w czasie pomiędzy pomiarami. Dzięki temu możliwe było otrzymanie dokładnego modelu, określonego parametrami:

x_1	r_x	r_{xy}	r_{xx}	y_1	r_y	r_{yx}	r_{yy}
1.1855	20.0975	-1.2041	-0.1876	0.2924	-2.6962	0.7880	-0.3097



Rysunek 4: Rozmiar populacji drapieżników i ofiar w zależności od czasu. Linia ciągłą przedstawiono pomiary rzeczywiste, przerywaną estymatę. Kolorem czerwonym oznaczono populację ofiar, niebieskim drapieżników.

7 Wykorzystane programy

Listing 1: Zadanie 1a

```
1 data = readtable('dane21.csv');
2 N1 = 20; N2 = 20; N3 = 20;
3 J = zeros(N1,N2,N3);
4 RX = linspace(-100,100,N2);
5 RXY = linspace(-1,1,N2);
6 RXX = linspace(-1,1,N3);
7 dt = mean(diff(data.t));
8 T = 0 : dt : 3;
9
10 for it1 = 1 : N1
11     for it2 = 1 : N2
12         for it3 = 1 : N3
13             rx = RX(it1);
14             rxy = RXY(it2);
15             rxx = RXX(it3);
16             x = zeros(size(T));
17             x(1) = data.x(1);
18             f = @(x,y) rx*x+rxy*x*y+rxx*x*x;
19             % Metoda Eulera
20             for i = 2 : length(T)
21                 x(i) = x(i-1) + dt * f(x(i-1), data.y(i-1));
22             end
23
24             % Adams-Bashforth
25             x(2) = x(1) + dt * f(x(1), data.y(1));
26             x(3) = x(2) + dt * f(x(2), data.y(2));
27             for i = 4 : length(T)
28                 x(i) = x(i-1) + dt/12 * (23*f(x(i-1), data.y(i-1))...
29                     -16*f(x(i-2), data.y(i-2)) +5*f(x(i-3), data.y(i-3)));
30             end
31
32             % Niejawna metoda Eulera
33             for i = 2 : length(T)
34                 %
35                 x1 = -(dt*rx - (dt^2*rx^2 + 2*dt^2*rx*rxy*data.y(i) + dt^2*rxy^2*data.y(i)^2 - 2*dt*rx - 2*dt*
36                     rxy*data.y(i) - 4*rxx*x(i-1)*dt + 1)^(1/2) + dt*rxy*data.y(i) - 1)/(2*dt*rxx);
37                 x2 = -(dt*rx + (dt^2*rx^2 + 2*dt^2*rx*rxy*data.y(i) + dt^2*rxy^2*data.y(i)^2 - 2*dt*rx - 2*dt*
38                     rxy*data.y(i) - 4*rxx*x(i-1)*dt + 1)^(1/2) + dt*rxy*data.y(i) - 1)/(2*dt*rxx);
39                 if isreal(x1) && isreal(x2) && (x1>0 || x2>0)
40                     x(i) = max(x1, x2);
41                 else
42                     x(i) = Inf;
43                     break
44                 end
45             end
46
47             J(it1, it2, it3) = sum((data.x'-x).*(data.x'-x));
48         end
49     end
50 [v, ind] = min(J(:));
51 [it1, it2, it3] = ind2sub(size(J), ind);
52 A = fminsearch(@(X) ApproxX(X(1), X(2), X(3)), [RX(it1), RXY(it2), RXX(it3)]);
53 fprintf("r_x=%f; r_xy=%f; r_xx=%f;\n", A(1), A(2), A(3));
54
55 function [J] = ApproxX(rx, rxy, rxx)
56 data = readtable('dane21.csv');
57 dt = mean(diff(data.t));
58 T = 0 : dt : 3;
59 x = zeros(1, length(T));
60 x(1) = data.x(1);
61
62 for i = 2 : length(T)
63     x1 = -(dt*rx - (dt^2*rx^2 + 2*dt^2*rx*rxy*data.y(i) + dt^2*rxy^2*data.y(i)^2 - 2*dt*rx - 2*dt*rxy*data.y(i) -
64         4*rxx*x(i-1)*dt + 1)^(1/2) + dt*rxy*data.y(i) - 1)/(2*dt*rxx);
65     x2 = -(dt*rx + (dt^2*rx^2 + 2*dt^2*rx*rxy*data.y(i) + dt^2*rxy^2*data.y(i)^2 - 2*dt*rx - 2*dt*rxy*data.y(i) -
66         4*rxx*x(i-1)*dt + 1)^(1/2) + dt*rxy*data.y(i) - 1)/(2*dt*rxx);
67     if isreal(x1) && isreal(x2) && (x1>0 || x2>0)
68         x(i) = max(x1, x2);
69     else
70         x(i) = Inf;
71         break
72     end
73 end
74 J = sum((data.x'-x(1,:)).*(data.x'-x(1,:)));
75 end % function
```

Listing 2: Zadanie 1b

```

1 data = readtable('dane21.csv');
2 N1 = 50;
3 N2 = 50;
4 N3 = 50;
5 J = zeros(N1,N2,N3);
6 RY = linspace(-100,100,N1);
7 RYX = linspace(-1,1,N2);
8 RYY = linspace(-1,1,N3);
9 dt = mean(diff(data.t));
10 T = 0 : dt : 3;
11 for it1 = 1 : N1
12     for it2 = 1 : N2
13         for it3 = 1 : N3
14             ry = RY(it1);
15             ryx = RYX(it2);
16             ryy = RYY(it3);
17             y = zeros(size(T));
18             y(1) = data.y(1);
19             f = @(x,y) ry*y+ryx*x*y+ryy*y*y;
20             % Metoda Eulera
21             % for i = 2 : length(T)
22             %     y(i) = y(i-1) + dt*f(data.x(i-1), y(i-1));
23             % end
24             % Metoda Adamsa-Bashfortha
25             y(2) = y(1) + dt * f(data.x(1), y(1));
26             y(3) = y(2) + dt * f(data.x(2), y(2));
27             for i = 4 : length(T)
28                 y(i) = y(i-1) + dt/12 * (23*f(data.x(i-1), y(i-1))...
29                     -16*f(data.x(i-2), y(i-2)) +5*f(data.x(i-3), y(i-3)));
30             end
31             % Niejawna metoda Eulera
32             % for i = 2 : length(T)
33             %     y1 = -(dt*ry - (dt^2*ry^2 + 2*dt^2*ry*ryx*data.x(i) + dt^2*ryx^2*data.x(i)^2 - 2*dt*ry - 2*dt*
34             % ryx*data.x(i) - 4*ryy*y(i-1)*dt + 1)^(1/2) + dt*ryx*data.x(i) - 1)/(2*dt*ryy);
35             %     y2 = -(dt*ry + (dt^2*ry^2 + 2*dt^2*ry*ryx*data.x(i) + dt^2*ryx^2*data.x(i)^2 - 2*dt*ry - 2*dt*
36             % ryx*data.x(i) - 4*ryy*y(i-1)*dt + 1)^(1/2) + dt*ryx*data.x(i) - 1)/(2*dt*ryy);
37             %     if isreal(y1) && isreal(y2) && (y1>0 || y2>0)
38             %         y(i) = max(y1, y2);
39             %     else
40             %         y(i) = Inf;
41             %         break
42             %     end
43             % end
44             J(it1, it2, it3) = sum((data.y'-y).*(data.y'-y));
45         end
46     end
47 end
48 end
49 [v, ind] = min(J(:));
50 [it1, it2, it3] = ind2sub(size(J), ind);
51 A = fminsearch(@(X) ApproxY(X(1), X(2), X(3)), [RY(it1), RYX(it2), RYY(it3)]);
52 fprintf("r_y=%f; r_yx=%f; r_yy=%f;\n", A(1), A(2), A(3));
53
54 function [J] = ApproxY(ry, ryx, ryy)
55 data = readtable('dane21.csv');
56 dt = mean(diff(data.t));
57 T = 0 : dt : 3;
58 y = zeros(size(T));
59 y(1) = data.y(1);
60 for i = 2 : length(T)
61     y1 = -(dt*ry - (dt^2*ry^2 + 2*dt^2*ry*ryx*data.x(i) + dt^2*ryx^2*data.x(i)^2 - 2*dt*ry - 2*dt*ryx*data.x(i) -
62         4*ryy*y(i-1)*dt + 1)^(1/2) + dt*ryx*data.x(i) - 1)/(2*dt*ryy);
63     y2 = -(dt*ry + (dt^2*ry^2 + 2*dt^2*ry*ryx*data.x(i) + dt^2*ryx^2*data.x(i)^2 - 2*dt*ry - 2*dt*ryx*data.x(i) -
64         4*ryy*y(i-1)*dt + 1)^(1/2) + dt*ryx*data.x(i) - 1)/(2*dt*ryy);
65     if isreal(y1) && isreal(y2) && (y1>0 || y2>0)
66         y(i) = max(y1, y2);
67     else
68         y(i) = Inf;
69         break
70     end
71 end
72 J = sum((data.y'-y).*(data.y'-y));
73 end % function

```


Listing 3: Zadanie 2

```

1 data = readtable('dane21.csv');
2 J = @(X) fun(X(1), X(2),X(3),X(4),X(5),X(6));
3
4 r_x=6.136346; r_xy=-0.069893; r_xx=-0.000000;
5 r_y=-5.884900; r_yx=0.059108; r_yy=-0.033141;
6
7 [W, val] = fminsearch(J, [r_x, r_y, r_xy, r_yx, r_xx, r_yy]);
8 disp(W);
9
10 f = figure;
11 hold on
12 f.Position = [100 100 1000 350];
13 plot(data.t, data.x, 'r');
14 plot(data.t, data.y, 'b');
15 f = @(t,x) [W(1)*x(1) + W(3)*x(1)*x(2)+ W(5)*x(1)*x(1); ...
16           W(2)*x(2) + W(4)*x(1)*x(2) + W(6)*x(2)*x(2)];
17 [t,y] = ode45(f, [0 3], [data.x(1), data.y(1)]);
18 plot(t,y(:,1), 'r--');
19 plot(t,y(:,2), 'b--');
20
21 function J = fun(rx, ry, rxy, ryx, rxx, ryy)
22     data = readtable('dane21.csv');
23
24     f = @(t, x) [rx*x(1) + rxy*x(1)*x(2)+rxx*x(1)*x(1); ...
25               ry*x(2) + ryx*x(1)*x(2) + ryy*x(2)*x(2)];
26
27     [t, y] = ode45(f, [0 3], [data.x(1), data.y(1)]);
28     X = interp1(t, y, data.t);
29     J = sum( (X-[data.x, data.y]).*(X-[data.x, data.y]), 'all');
30 end % function

```

Listing 4: Zadanie 3

```

1 data = readtable('Chromista.csv');
2 data = renamevars(data, data.Properties.VariableNames, ["t", "x", "y"]);
3 data.t = normalize(data.t, 'range');
4 SCALING = 5;
5
6 N1 = 20; N2 = 20; N3 = 20; N4 = 20;
7 J = zeros(N1,N2,N3, N4);
8 RX = linspace(0,50,N1);
9 RXY = linspace(-3,0,N2);
10 RXX = linspace(-1,1,N3);
11 X = linspace(min(data.x),max(data.x),N4);
12 dt = mean(diff(data.t))/SCALING;
13 T = 0 : dt : 1;
14 YInterp = interp1(data.t, data.y, T);
15 XInterp = interp1(data.t, data.x, T);
16
17 for it1 = 1 : N1
18     for it2 = 1 : N2
19         for it3 = 1 : N3
20             for it4 = 1 : N4
21                 rx = RX(it1);
22                 rxy = RXY(it2);
23                 rxx = RXX(it3);
24                 x = zeros(size(T));
25                 x(1) = X(it4);
26                 f = @(x,y) rx*x+rxy*x*y+rxx*x*x;
27                 x(2) = x(1) + dt * f(x(1), YInterp(1));
28                 x(3) = x(2) + dt * f(x(2), YInterp(2));
29                 for i = 4 : length(T)
30                     x(i) = x(i-1) + dt/12 * (23*f(x(i-1), YInterp(i-1))...
31                               -16*f(x(i-2), YInterp(i-2)) +5*f(x(i-3), YInterp(i-3)));
32                 end
33                 J(it1, it2, it3, it4) = sum((XInterp-x).*(XInterp-x));
34             end
35         end
36     end
37 end
38 [~, ind] = min(J(:));
39 [it1, it2, it3, it4] = ind2sub(size(J), ind);
40 A = fminsearch(@(X) ApproxX(X(1), X(2), X(3), X(4)), [RX(it1), RXY(it2), RXX(it3), X(it4)], optimset('Display','off'));
41 r_x=A(1); r_xy=A(2); r_xx=A(3); x0=A(4);
42
43 J = zeros(N1,N2,N3, N4);
44 RY = linspace(-30,0,N1);
45 RYX = linspace(0,3,N2);
46 RYY = linspace(-1,1,N3);
47 Y = linspace(min(data.y),max(data.y),N4);
48 for it1 = 1 : N1
49     for it2 = 1 : N2
50         for it3 = 1 : N3
51             for it4 = 1 : N4
52                 ry = RY(it1);
53                 ryx = RYX(it2);
54                 ryy = RYY(it3);
55                 y = zeros(size(T));
56                 y(1) = Y(it4);
57                 f = @(x,y) ry*y+ryx*x*y+ryy*y*y;
58                 y(2) = y(1) + dt * f(data.x(1), y(1));
59                 y(3) = y(2) + dt * f(data.x(2), y(2));

```

```

60         for i = 4 : length(T)
61             y(i) = y(i-1) + dt/12 * (23*f(XInterp(i-1), y(i-1))...
62                 -16*f(XInterp(i-2), y(i-2)) +5*f(XInterp(i-3), y(i-3)));
63         end
64         J(it1, it2, it3, it4) = sum((YInterp-y).*(YInterp-y));
65     end
66 end
67 end
68 end
69 [v, ind] = min(J(:));
70 [it1, it2, it3, it4] = ind2sub(size(J), ind);
71 A = fminsearch(@(X) ApproxY(X(1), X(2), X(3), X(4)), [RY(it1), RYX(it2), RYY(it3), Y(it4)], optimset('Display', 'off'));
72 r_y=A(1); r_yx=A(2); r_yy=A(3); y0=A(4);
73
74 J = @(X) fun3(X(1), X(2), X(3), X(4), X(5), X(6), X(7), X(8));
75 W = fminsearch(J, [x0, y0, r_x, r_y, r_xy, r_yx, r_xx, r_yy], optimset('Display', 'off'));
76 disp(W)
77
78 f=figure;
79 hold on
80 f.Position = [100 100 1000 350];
81 plot(data.t, data.x, 'r');
82 plot(data.t, data.y, 'b');
83 f = @(t,x) [W(3)*x(1) + W(5)*x(1)*x(2) + W(7)*x(1)*x(1); ...
84             W(4)*x(2) + W(6)*x(1)*x(2) + W(8)*x(2)*x(2)];
85 [t,y] = ode45(f, [0 1], [W(1), W(2)]);
86 plot(t,y(:,1), 'r--');
87 plot(t,y(:,2), 'b--');
88
89 function [J] = ApproxX(rx, rxy, rxx, x0)
90 data = readtable('Chromista.csv');
91 data = renamevars(data, data.Properties.VariableNames, ["t", "x", "y"]);
92 SCALING = 5;
93 dt = mean(diff(data.t))/SCALING;
94 T = 0:dt:1;
95 YInterp = interp1(data.t, data.y, T);
96 XInterp = interp1(data.t, data.x, T);
97 data.t = normalize(data.t, 'range');
98
99 x = zeros(1, length(T));
100 x(1) = x0;
101 f = @(x,y) rx*x+rxy*x*y+rxx*x*x;
102 x(2) = x(1) + dt * f(x(1), YInterp(1));
103 x(3) = x(2) + dt * f(x(2), YInterp(2));
104 for i = 4 : length(T)
105     x(i) = x(i-1) + dt/12 * (23*f(x(i-1), YInterp(i-1))...
106         -16*f(x(i-2), YInterp(i-2)) +5*f(x(i-3), YInterp(i-3)));
107 end
108 J = sum((XInterp-x).*(XInterp-x));
109 end % function
110
111 function [J] = ApproxY(ry, ryx, ryy, y0)
112 data = readtable('Chromista.csv');
113 data = renamevars(data, data.Properties.VariableNames, ["t", "x", "y"]);
114 SCALING = 5;
115 dt = mean(diff(data.t))/SCALING;
116 T = 0:dt:1;
117 YInterp = interp1(data.t, data.y, T);
118 XInterp = interp1(data.t, data.x, T);
119 data.t = normalize(data.t, 'range');
120 dt = mean(diff(data.t));
121 y = zeros(1, length(T));
122 y(1) = y0;
123 f = @(x,y) ry*y+ryx*x*y+ryy*y*y;
124 y(2) = y(1) + dt * f(data.x(1), y(1));
125 y(3) = y(2) + dt * f(data.x(2), y(2));
126 for i = 4 : length(T)
127     y(i) = y(i-1) + dt/12 * (23*f(XInterp(i-1), y(i-1))...
128         -16*f(XInterp(i-2), y(i-2)) +5*f(XInterp(i-3), y(i-3)));
129 end
130 J = sum((YInterp-y).*(YInterp-y));
131 end % function
132
133 function J = fun3(x1, y1, rx, ry, rxy, ryx, rxx, ryy)
134 data = readtable('Chromista.csv');
135 data = renamevars(data, data.Properties.VariableNames, ["t", "x", "y"]);
136 data.t = normalize(data.t, 'range');
137 f = @(t, x) [rx*x(1) + rxy*x(1)*x(2)+rxx*x(1)*x(1); ...
138             ry*x(2) + ryx*x(1)*x(2) + ryy*x(2)*x(2)];
139 [t, y] = ode45(f, [0 1], [x1, y1]);
140 X = interp1(t, y, data.t);
141 J = sum((X-[data.x, data.y]).*(X-[data.x, data.y]), 'all');
142 end % function

```