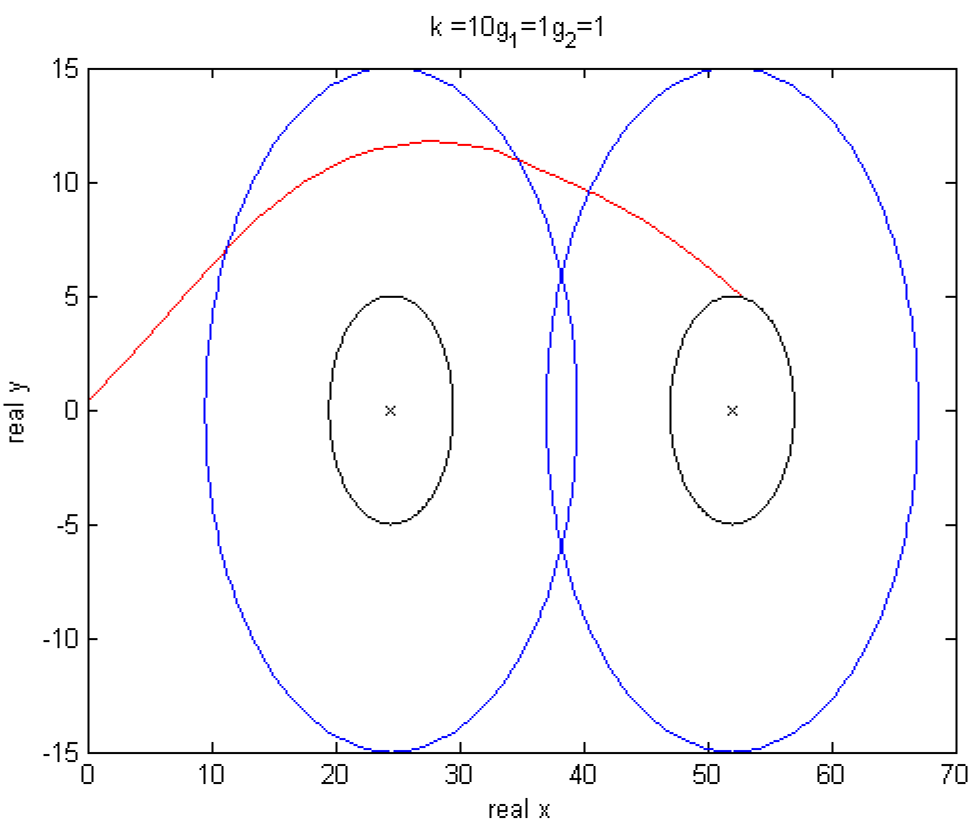
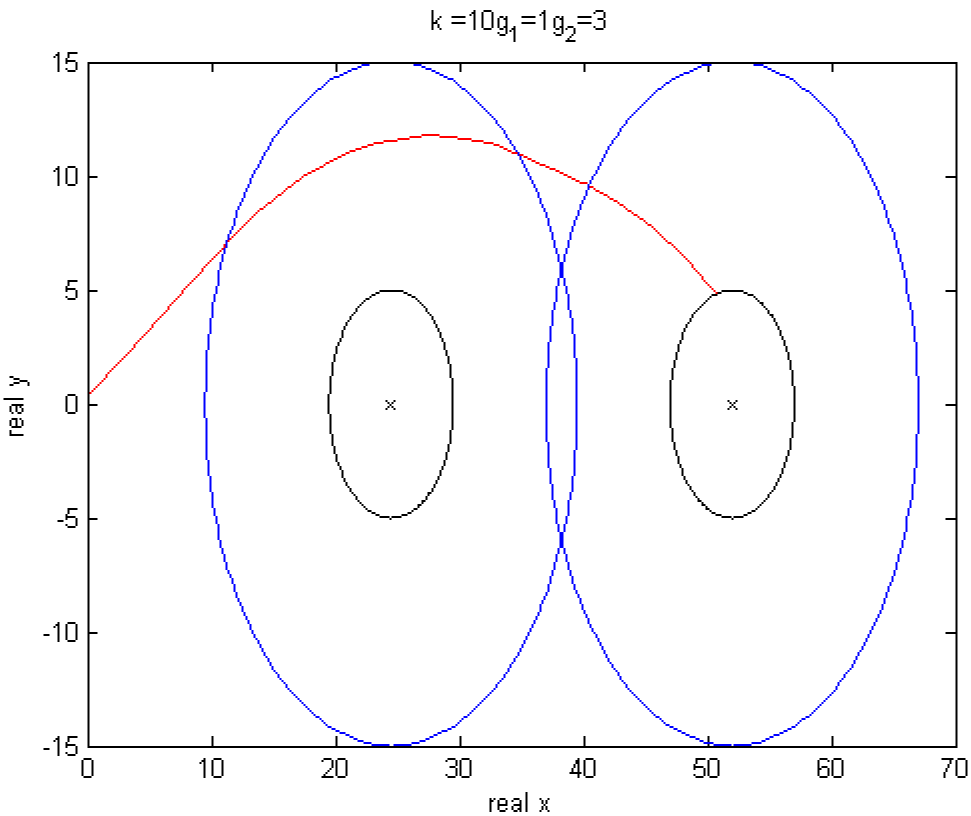
```
function struct =  AngryBirdsDriverLittleG
%created by Zach Mink and E. Backus April 2012
%all lengths are supposed to be measured in red-bird diameters
%note: coordinates are in approximate red bird lengths-units of measure
%speed of red bird initially was recorded shortly after launch(probably a
%bit small)
%data is probably taken in the software at every frame - given in units of
%per second
close all

%constants
    Asteroid1 = [24.5, 0]; %position on screen
    x1 = Asteroid1(1);
    y1 = Asteroid1(2);
    R1 = 5;                 %asteroid radius
    fR1 = 15;               %field radius
    Asteroid2 = [52, 0];    %position on screen
    x2 = Asteroid2(1);
    y2 = Asteroid2(2);
    R2 = 5;                  %asteroid 2 radius
    fR2 = 15;                %field radius
    %RedBird = [0, 32.90362639,.3448, 6.568309441];
    RedBird = [0, 4.5,.3448, 2.7];
    xb0 = RedBird(1);
    vxb0 = RedBird(2);
    yb0 = RedBird(3);
    vyb0 = RedBird(4);
    mBird = 100;             %mass of a red bird
    i = 0;

for     k = [10 50 100];            %drag coefficient
    for g1 = [1 3 10];              %gravitational constant
        for g2 = [1 3 10];          %second asteroid gravitational constant
          i = i + 1;
          constants = [x1, y1, x2, y2, g1, g2, k, mBird, fR1, fR2, R1, R2];

        %characteristic length and time
          xc = x1;
          tau = sqrt(x1/g1);

            %%set the options function of the ode solver to
            %options = odeset('Events',@eventsLocationDetection);

        %solve the coupled ODEs
        %specify the initial position and speed in unitless terms
        init = [xb0./x1, vxb0./(x1/tau), yb0./x1, vyb0./(x1/tau)];
        event = 65;
        tSpan = [0 event/(tau)];%convert desired time into unitless time

        options = odeset('Events',@eventsCollisionDetection);

        %struct = ode45(@fAngryBirdDiffEQ, tSpan, init, options,constants);
        struct = ode23s(@fAngryBirdDiffEQ, tSpan, init, options,constants);
        %struct = ode15s(@fAngryBirdDiffEQ, tSpan, init,options,constants);
        %struct = ode113(@fAngryBirdDiffEQ, tSpan, init,options,constants);
        %struct = ode23t(@fAngryBirdDiffEQ, tSpan, init,options,constants);
        %struct = ode23tb(@fAngryBirdDiffEQ, tSpan,init,options,constants);

        %catch unitless output
            ut = struct.x;        %struct.x is unitless time T
            ux = struct.y(1,:);
            uvx = struct.y(2,:);
            uy = struct.y(3,:);
```

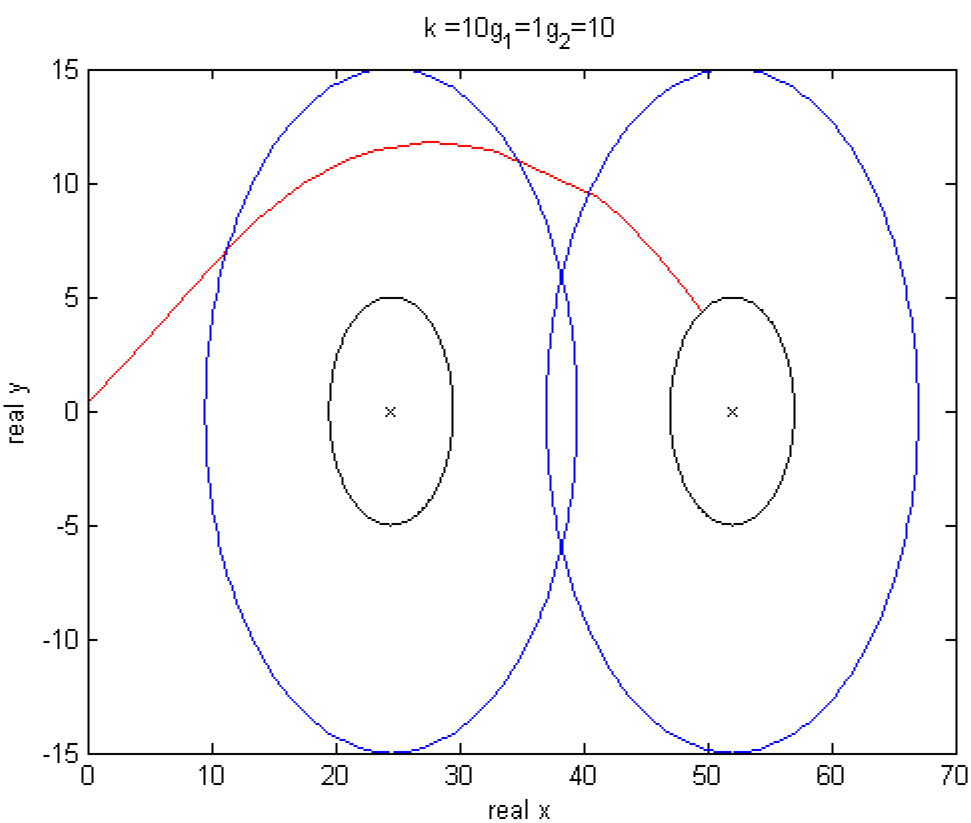
```
        uvy = struct.y(4,:);

    % %plot unitless variables
    %         subplot(3,2,1)%positions versus time
    %             plot(ut,ux,'r',ut,uy,'b')
    %             legend('unitless x', 'unitless y')
    %             xlabel('unitless time')
    %         subplot(3,2,3) %velocities versus time
    %             plot(ut,uvx,'r',ut,uvy,'b')
    %             legend('unitless vx', 'unitless vy')
    %             xlabel('unitless time')
    %         subplot(3,2,5)%positions in space
    %             plot(ux,uy,'r',x1/xc,y1/xc,'kx',x2/xc,y2/xc,'kx')
    %             xlabel('unitless x')
    %             ylabel('unitless y')
    %
    % %plot real variables
    %         realTime = ut*tau;
    %         realX = ux*xc;
    %         realY = uy*xc;
    %         realVx = uvx*xc/tau;
    %         realVy = uvy*xc/tau;
    %         subplot(3,2,2)%positions versus time
    %             plot(realTime,realX,'r',realTime,realY,'b')
    %             legend('real x', 'real y')
    %             xlabel('real time')
    %         subplot(3,2,4) %velocities versus time
    %             plot(realTime,realVx,'r',realTime,realVy,'b')
    %             legend('real vx', 'real vy')
    %             xlabel('real time')
    %         subplot(3,2,6)%positions in space
    %             plot(realX,realY,'r',x1,y1,'kx',x2,y2,'kx')
    %             %axis([0 80 -40 40])
    %             xlabel('real x')
    %             ylabel('real y')
    %

    %plot real variables
            figure(i)
            realX = ux*xc;
            realY = uy*xc;
            %positions in space
                    %define coordinates for gravitational circles and
                    %asteroid circles
                    N = 256;
                    theta = (0:N)*2*pi/N;
                    %Note: using the same radius for both gravity
                    %fields and asteroids
                    x = fR1*cos(theta) + x1;
                    y = fR1*sin(theta) + y1;
                    xA = R1*cos(theta) + x1;
                    yA = R1*sin(theta) + y1;

    %plot(realX,realY,'r-',x1,y1,'kx',x2,y2,'kx')
    plot(realX,realY,'r-',x1,y1,'kx',x2,y2,'kx',x,y,'b-',x+(x2-x1), ...
        y+(y2-y1),'b-',xA,yA,'k', xA+(x2-x1),yA + (y2-y1), 'k')
    %axis([0 80 -40 40])
    xlabel('real x')
    ylabel('real y')
    title(strcat('k =',num2str(k),'g_{1}=',num2str(g1),'g_{2}=',...
        num2str(g2)));
    end
end
end
```

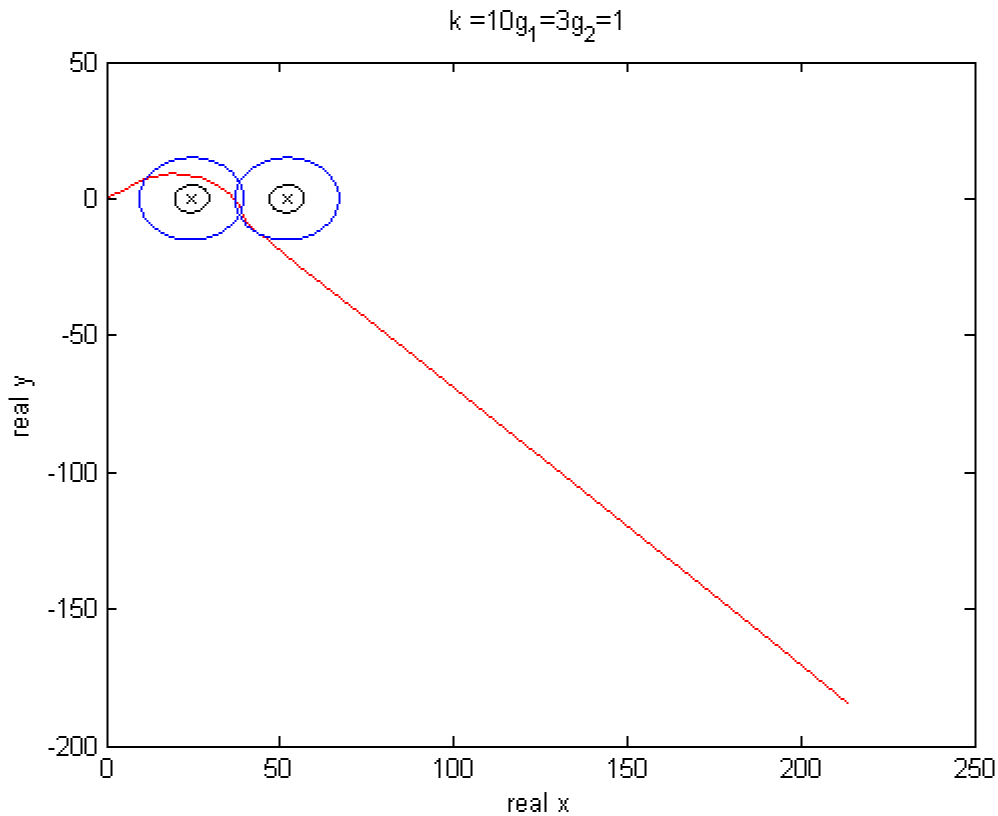
```
end

function vPrime = fAngryBirdDiffEQ(~, v, constants)
%constants
    x1 = constants(1);
    y1 = constants(2);
    x2 = constants(3);
    y2 = constants(4);
    g1 = constants(5);
    g2 = constants(6);
    k = constants(7);
    mBird = constants(8);
    fR1 = constants(9);
    fR2 = constants(10);

   A = 1;%boolean coefficient for first asteroid
   B = 1;%boolean coefficient for second asteroid
   C = 0;%boolean drag coefficient to turn on drag when inside either field

%initialize vPrime
    vPrime = zeros(4,1);

%If the bird is outside of either asteroid field, remove the effects of
%that asteroid field
    if fR1^2 < (v(1)*x1 - x1)^2 + (v(3)*x1 - y1)^2%outside asteriod1 field
        A = 0;
    end

    if fR2^2 < (v(1)*x1 - x2)^2 + (v(3)*x1 - y2)^2%outside asteriod2 field
        B = 0;
    end

%If the bird is inside either asteroid field, include drag
   if (A == 1 || B == 1 )
       C = 1;
   end

    %differential equations of motion
    vPrime(1) = v(2);
    vPrime(2) = A*(1-v(1))./sqrt((1-v(1)).^2 + ((y1/x1)-v(3)).^2)...
                + B*(g2/g1)*(x2/x1 - v(1))./sqrt(((x2/x1)-v(1)).^2 + ...
                ((y2/x1)-v(3)).^2)...
                - C*k/(g1*mBird) * v(2)./sqrt(v(2).^2 + v(4).^2);
    vPrime(3) = v(4);
    vPrime(4) = A*((y1/x1)-v(3))./sqrt((1-v(1)).^2 + ((y1/x1)-v(3)).^2)...
                + B*(g2/g1)*(y2/x1 - v(3))./sqrt(((x2/x1)-v(1)).^2 + ...
                ((y2/x1)-v(3)).^2)...
                - C*k/(g1*mBird) * v(4)./sqrt(v(2).^2 + v(4).^2);
end



%determines if bird has collided with an asteroid, if so, terminates
%the ode solver
function [value,isTerminal,dir]=eventsCollisionDetection(~,v,constants)

    %acquire all of the constants needed for this function
    x1 = constants(1);
    y1 = constants(2);
    x2 = constants(3);
    y2 = constants(4);
    R1 = constants(11);
    R2 = constants(12);

    %define the variables if neither if statement is true
```

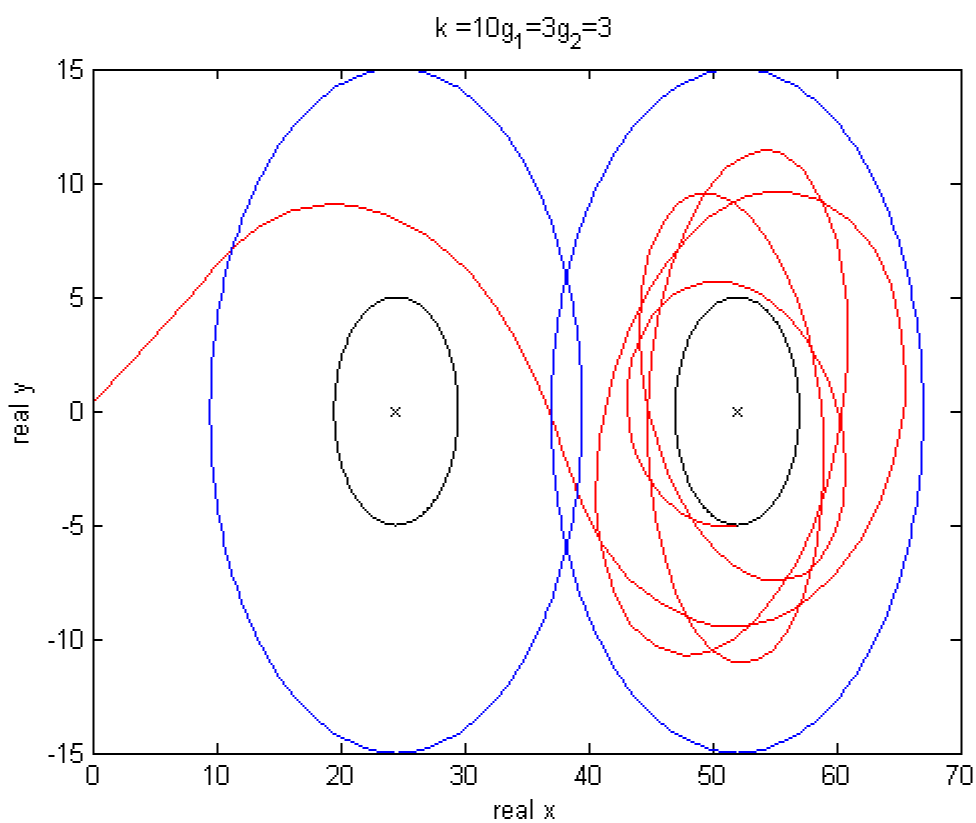
```
    value = 1;
    isTerminal = 0;
    dir = 0;

     if R1^2 > (v(1)*x1 - x1)^2 + (v(3)*x1 - y1)^2%bird is inside asteriod1
        value = 0;
        isTerminal = 1;
        dir = 0;
     end

     if R2^2 > (v(1)*x1 - x2)^2 + (v(3)*x1 - y2)^2%bird is inside asteriod2
        value = 0;
        isTerminal = 1;
        dir = 0;
     end

end



ans =

     solver: 'ode23s'
    extdata: [1x1 struct]
          x: [1x27 double]
          y: [4x27 double]
         xe: 2.4825
         ye: [4x1 double]
         ie: 1
      stats: [1x1 struct]
      idata: [1x1 struct]
```

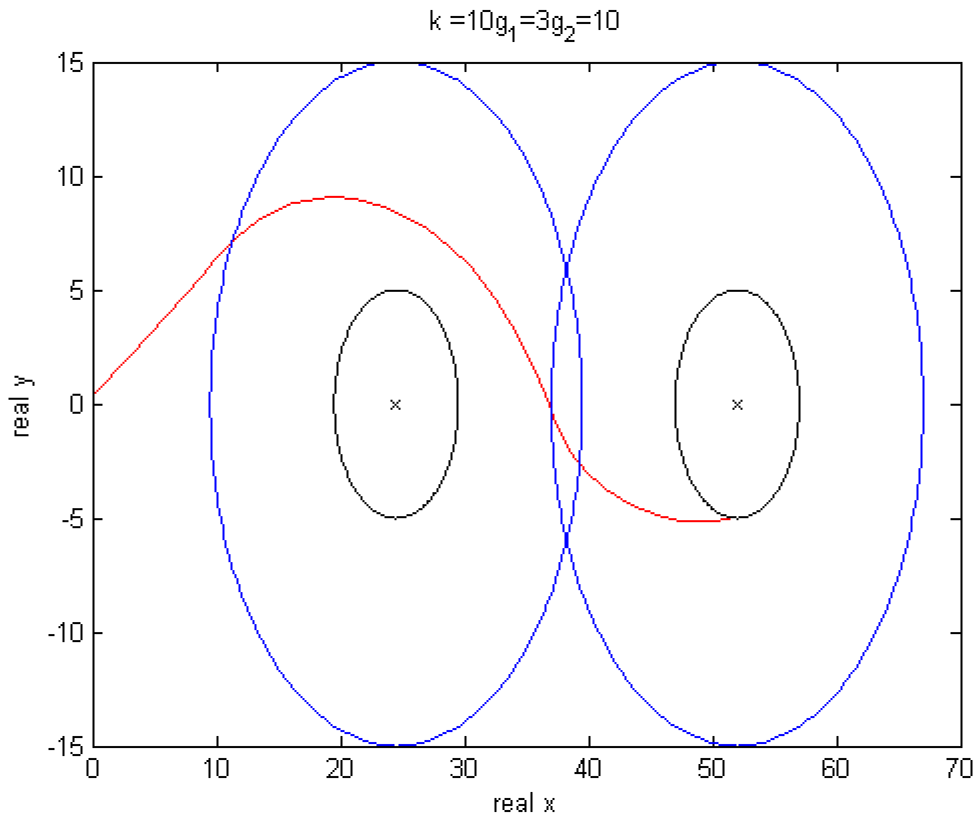



k =10g1=1g2=3
real y
real x

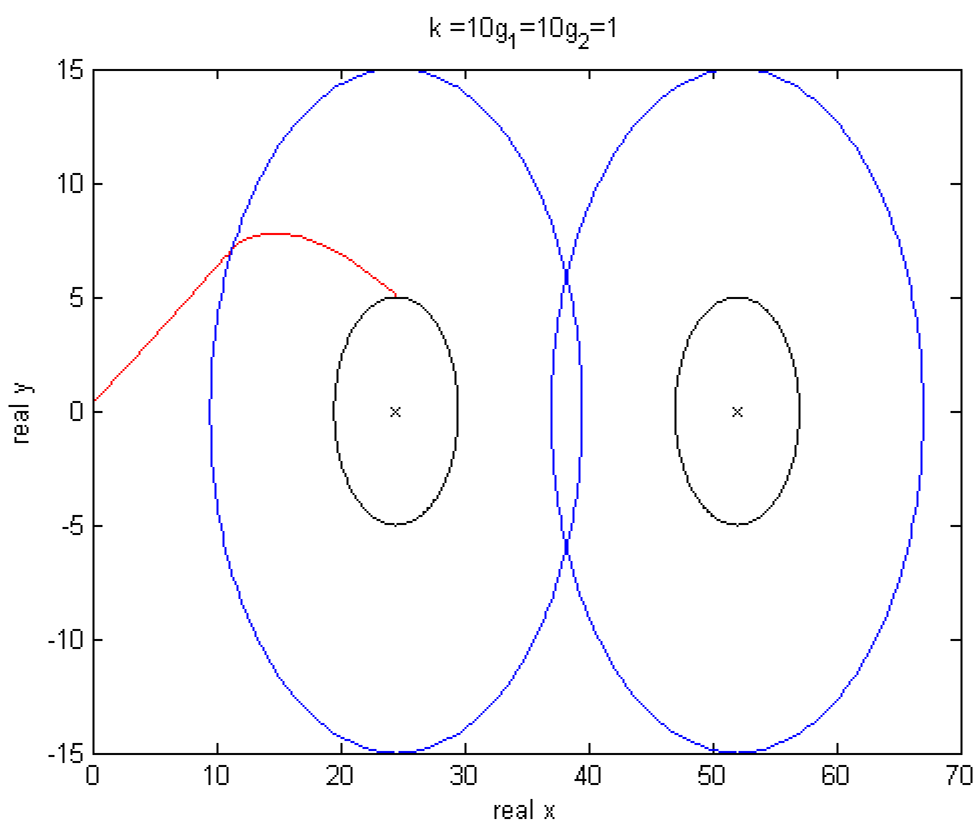

k =10g1=1g2=10
real y
real x


k =10g$_1$=3g$_2$=1
real y
real x

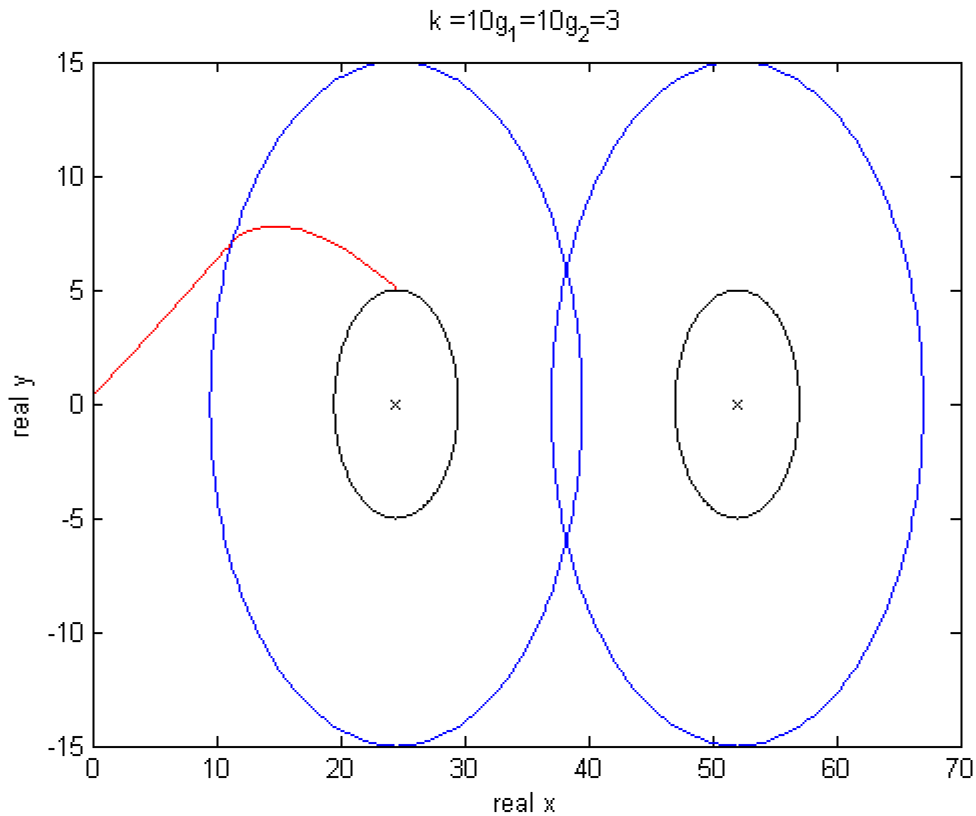

k =10g$_1$=3g$_2$=3
real y
real x

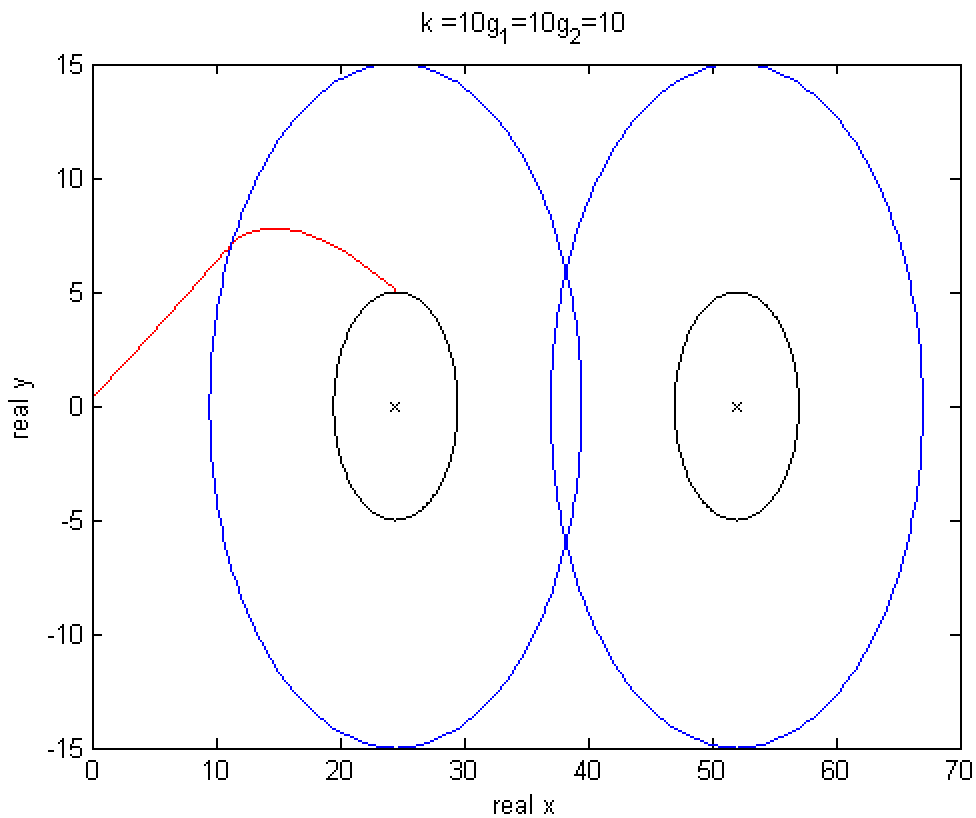


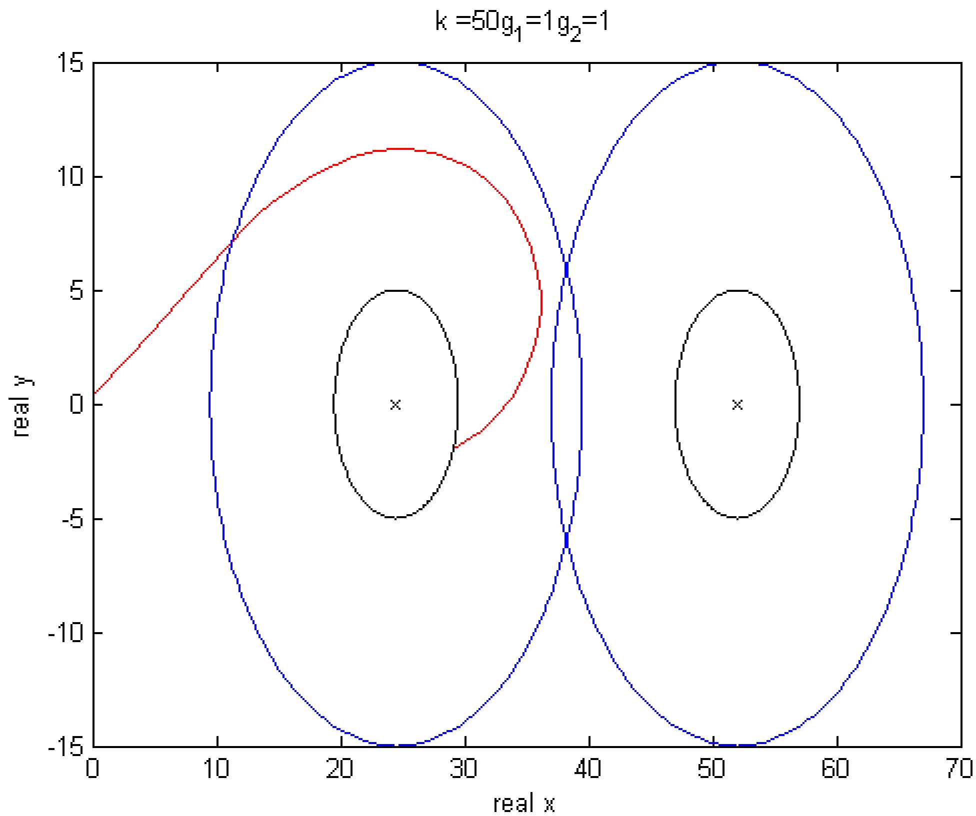
k =10g$_1$=3g$_2$=10
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70

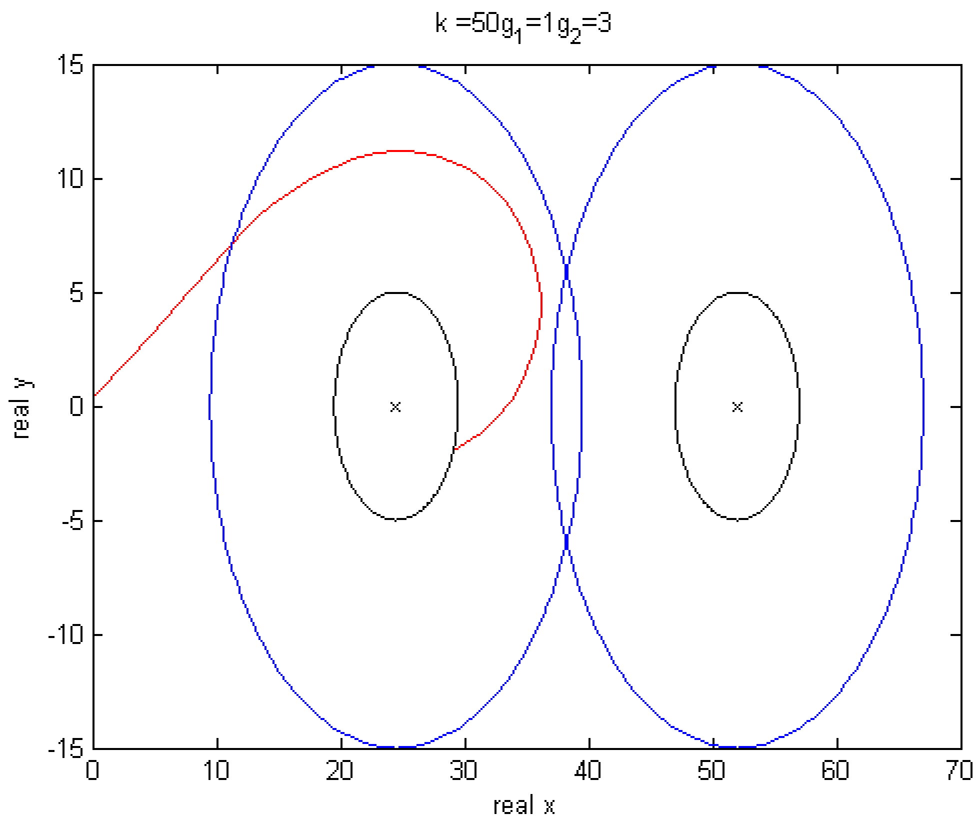


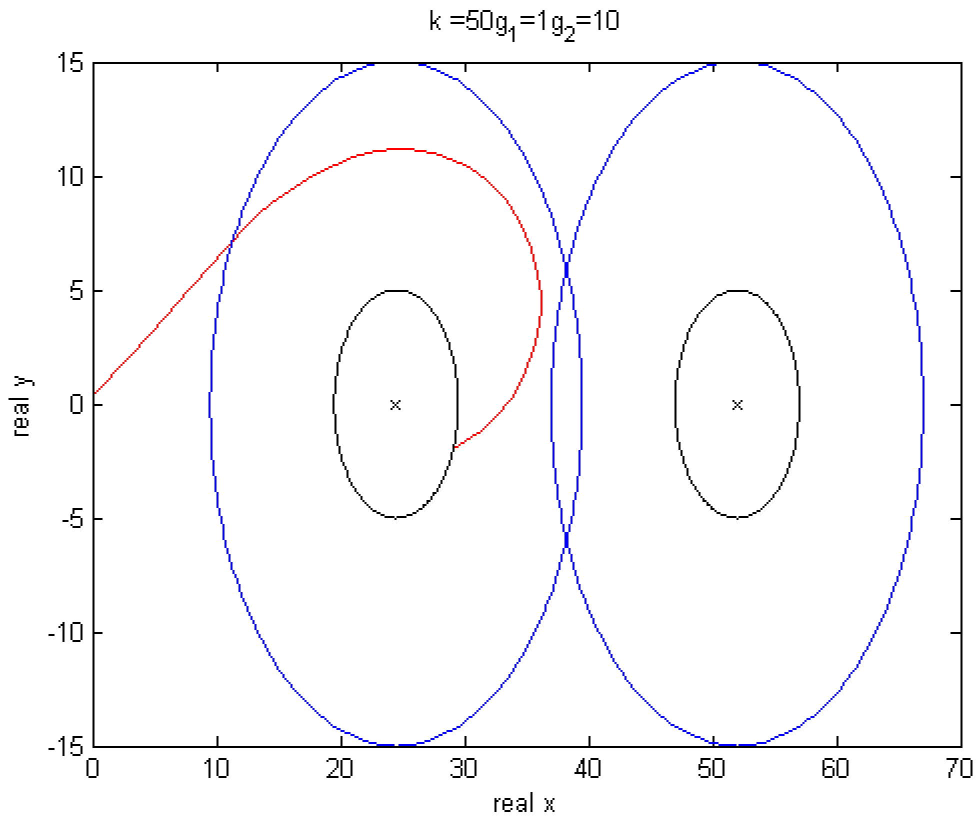
k =10g$_1$=10g$_2$=1
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70

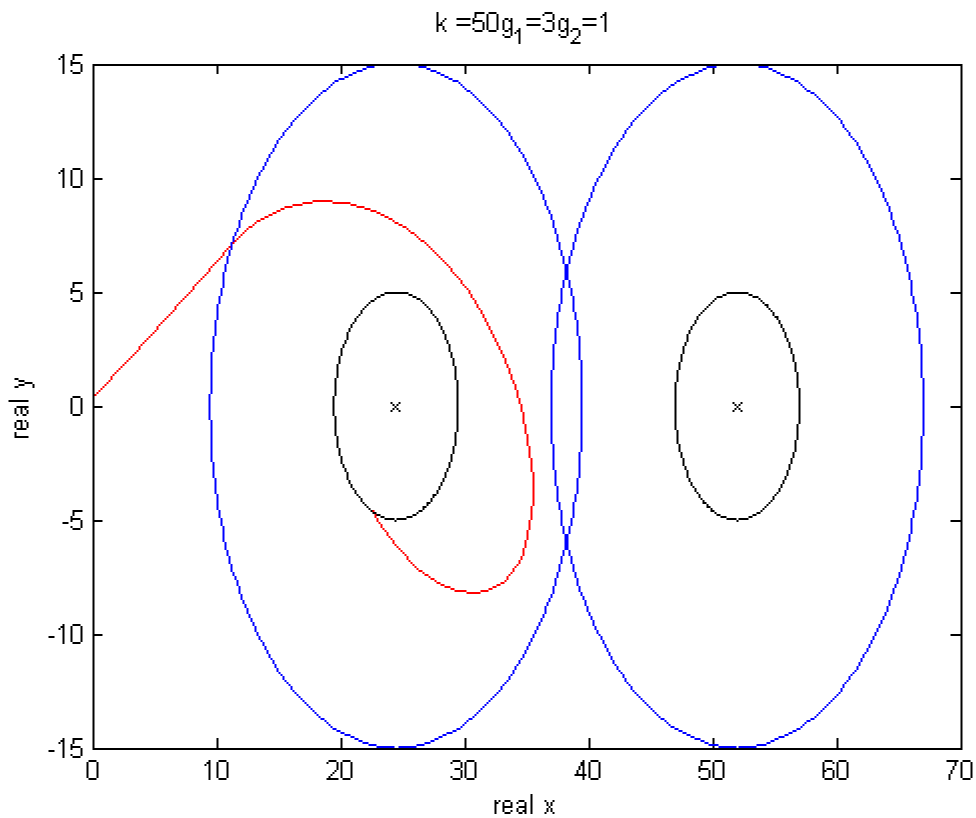

k =10g1=10g2=3
15
10
5
0
-5
-10
-15
real y
0
10
20
30
40
50
60
70
real x


k =10g1=10g2=10
15
10
5
0
-5
-10
-15
real y
0
10
20
30
40
50
60
70
real x

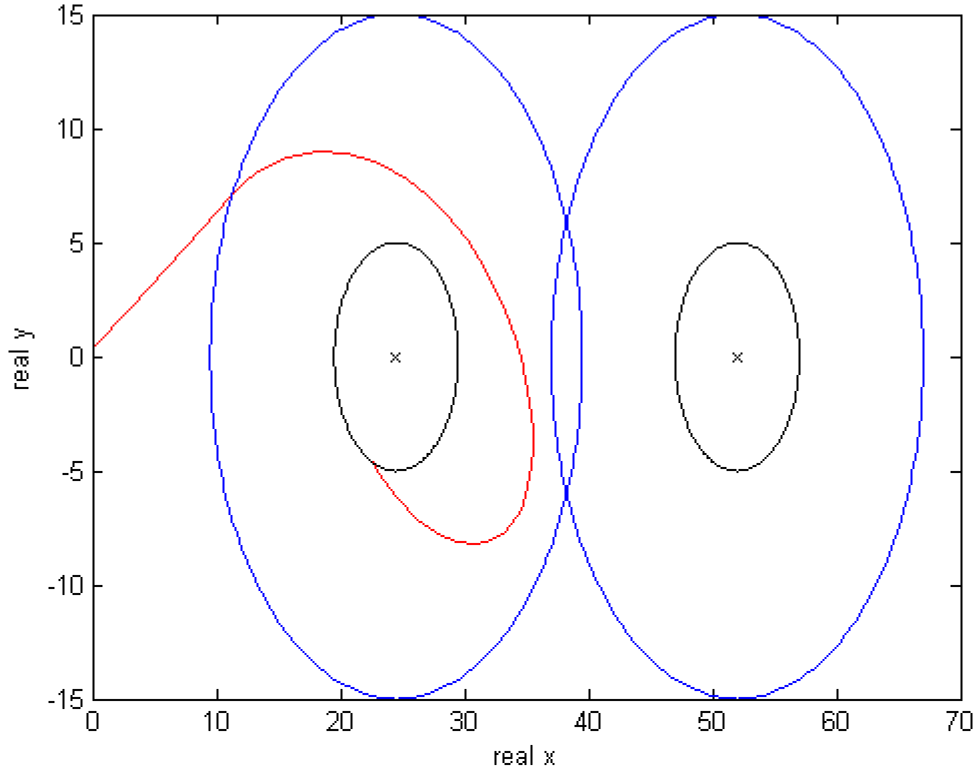

k =50g1=1g2=1
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70


k =50g1=1g2=3
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70

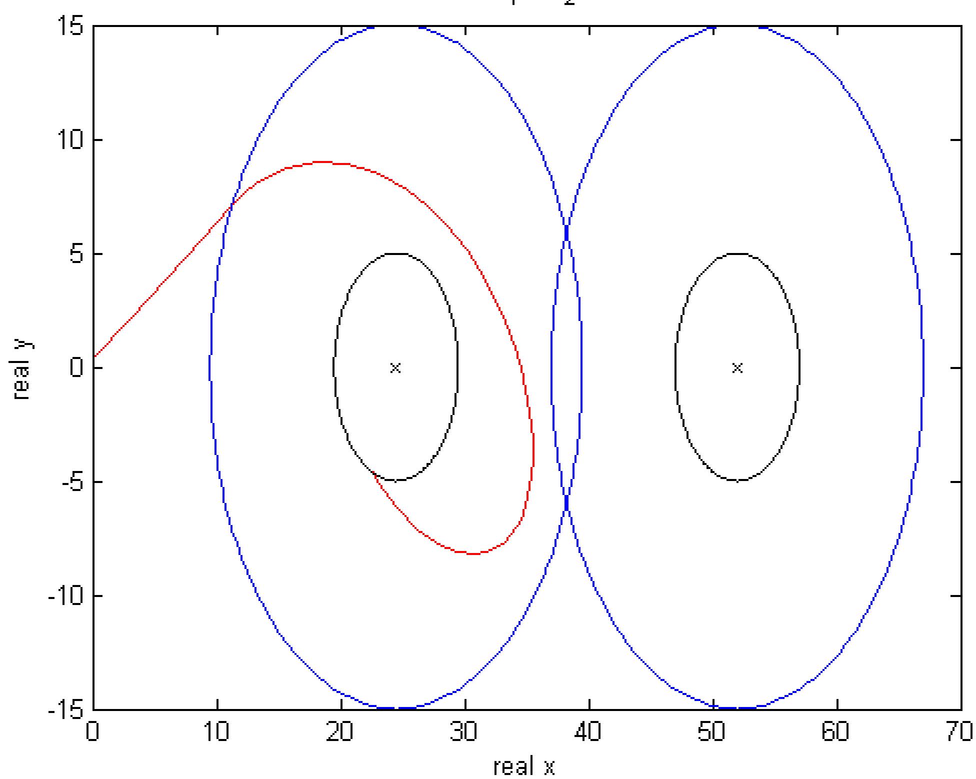

k =50g1=1g2=10
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70

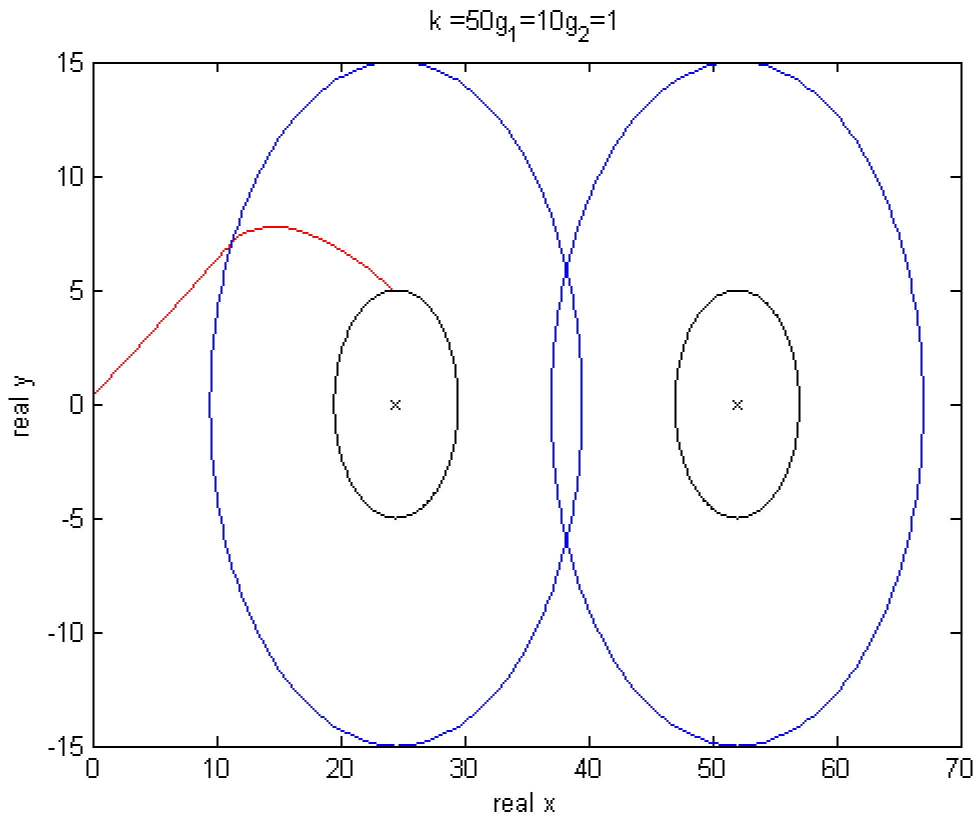

k =50g1=3g2=1
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70

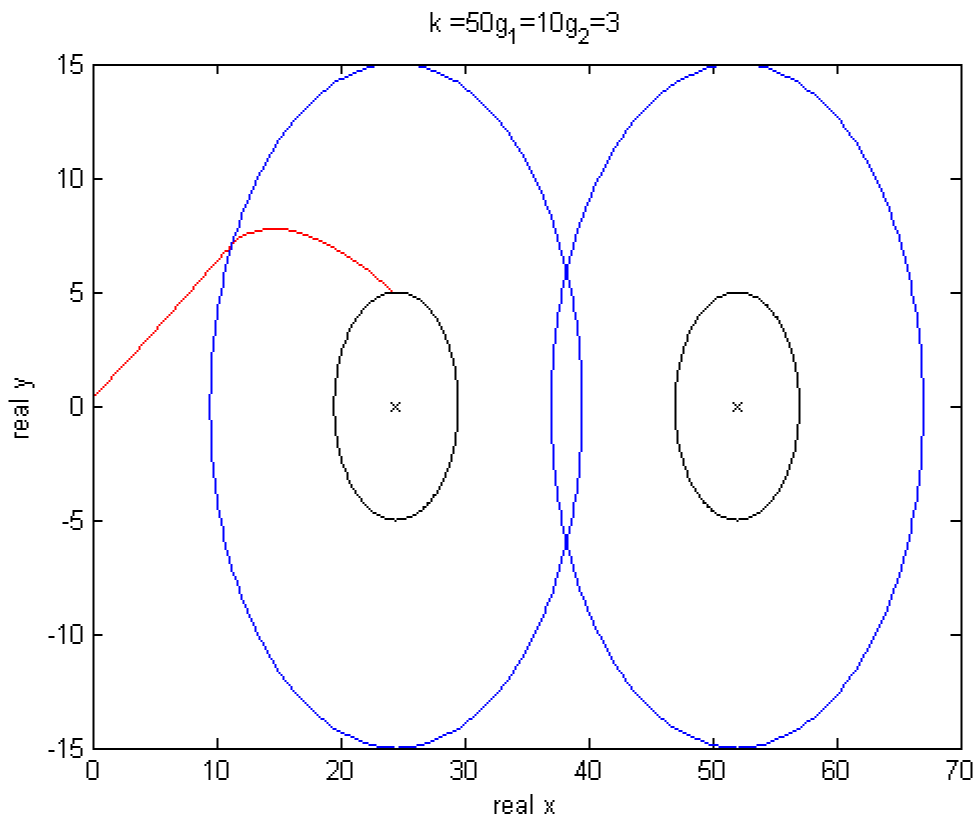
k =50g$_1$=3g$_2$=3

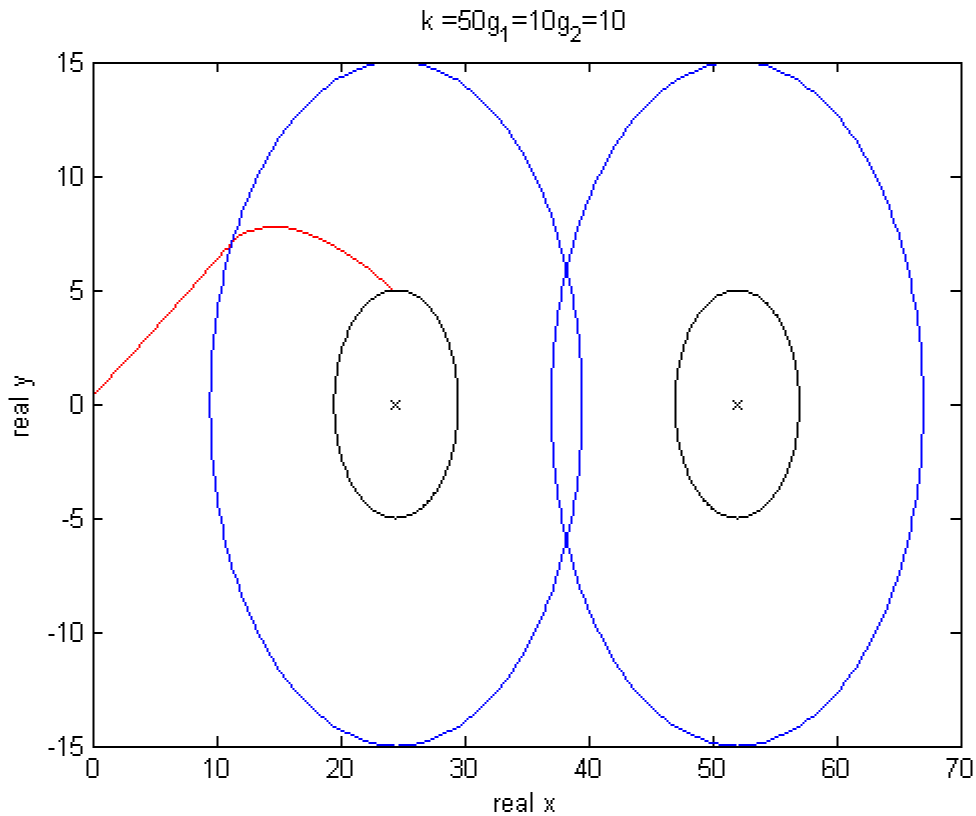


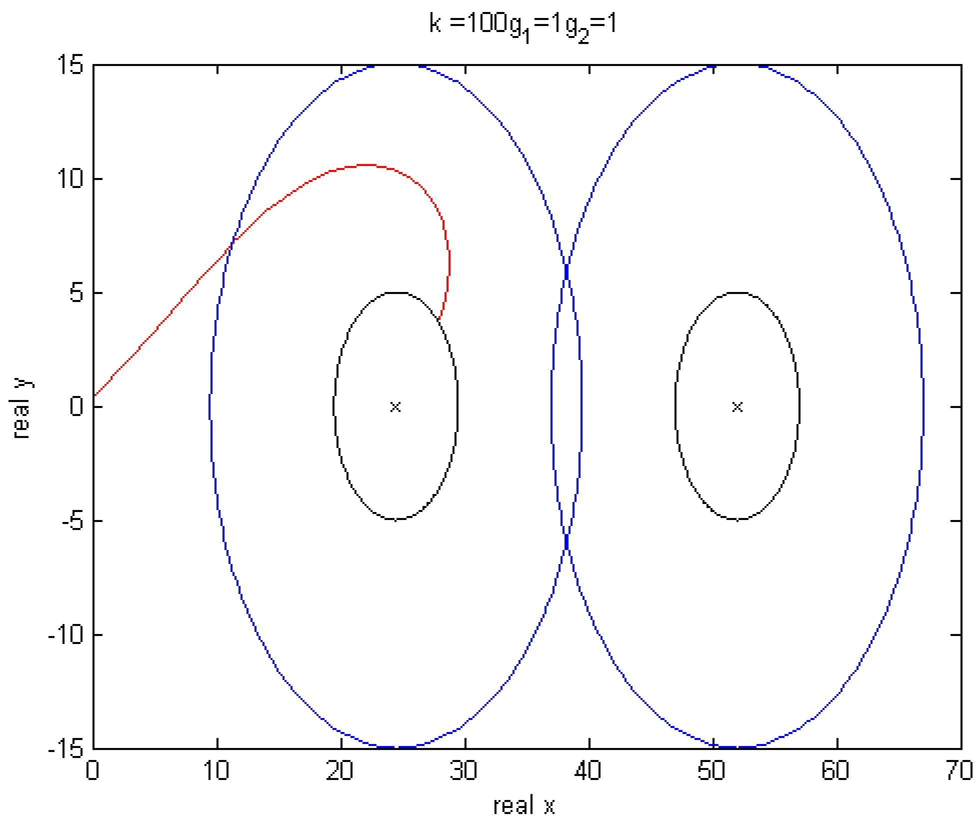
k =50g$_1$=3g$_2$=10

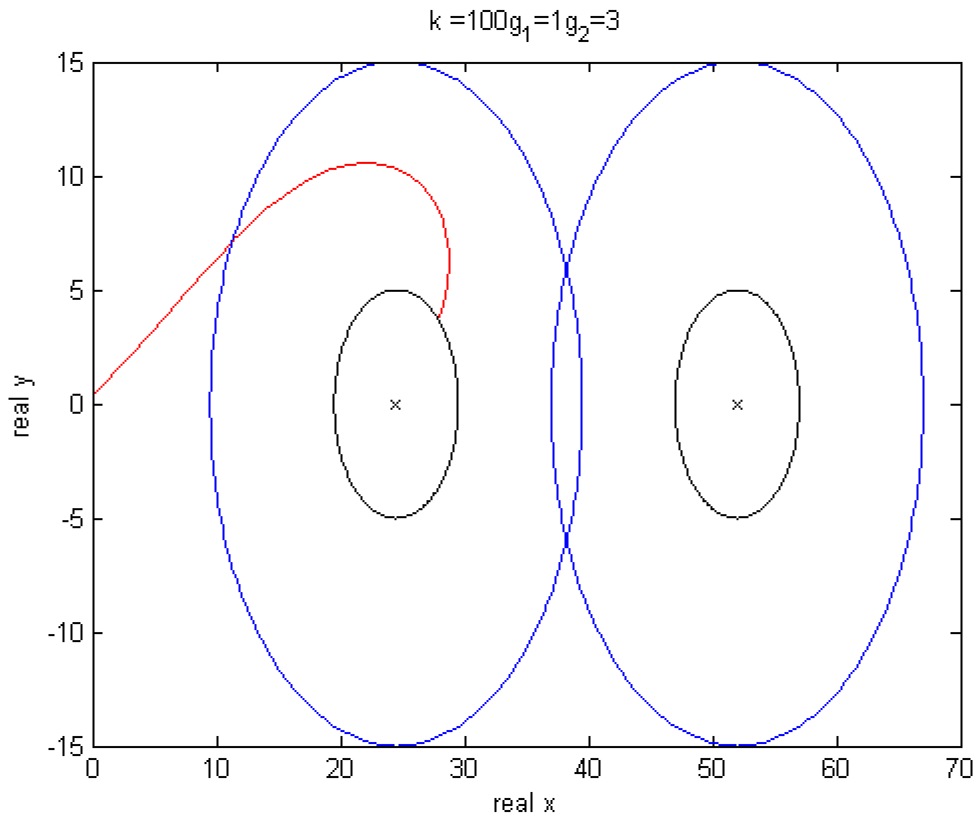


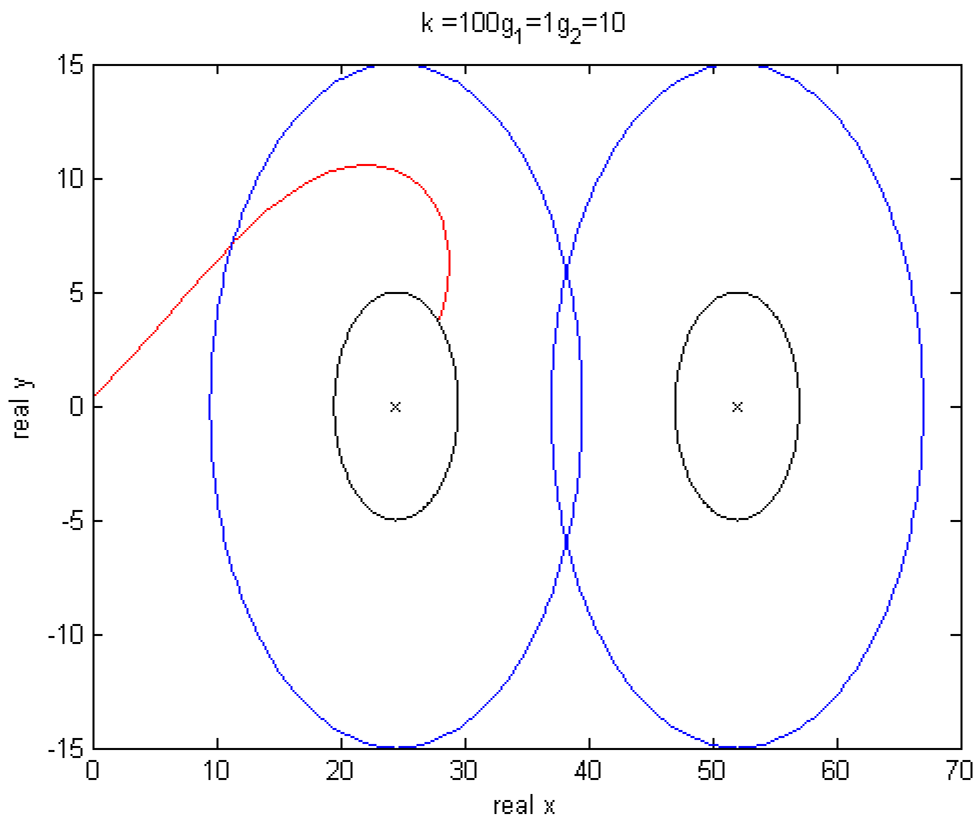

k =50g1=10g2=1
real y
real x

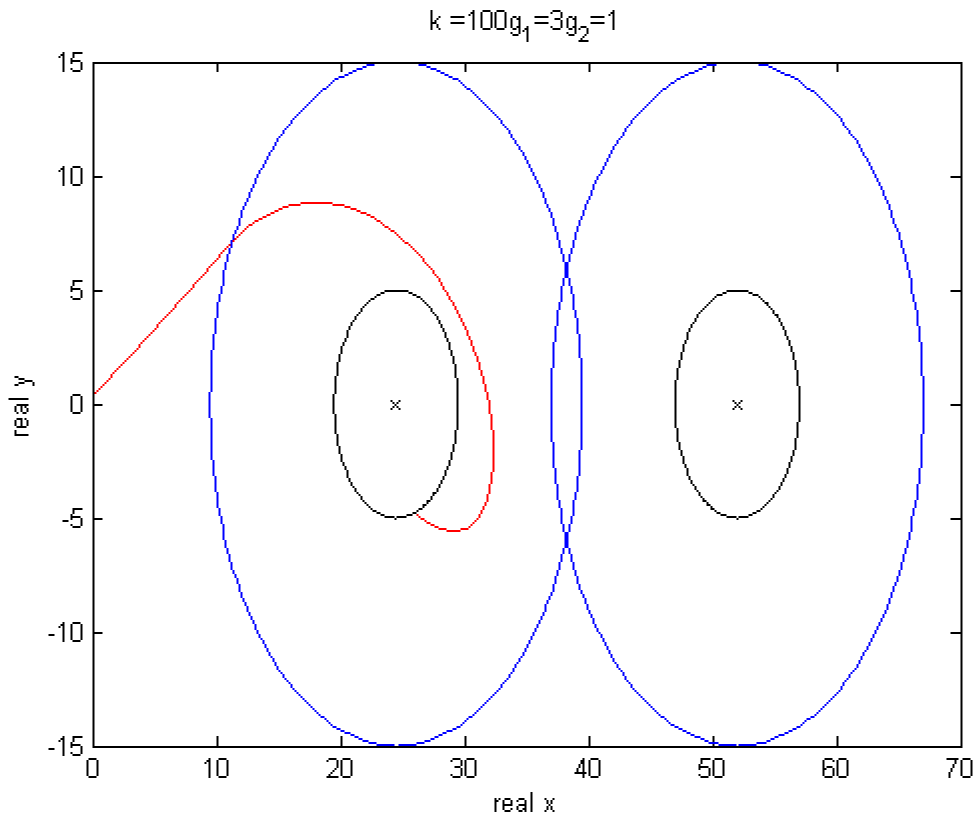

k =50g1=10g2=3
real y
real x

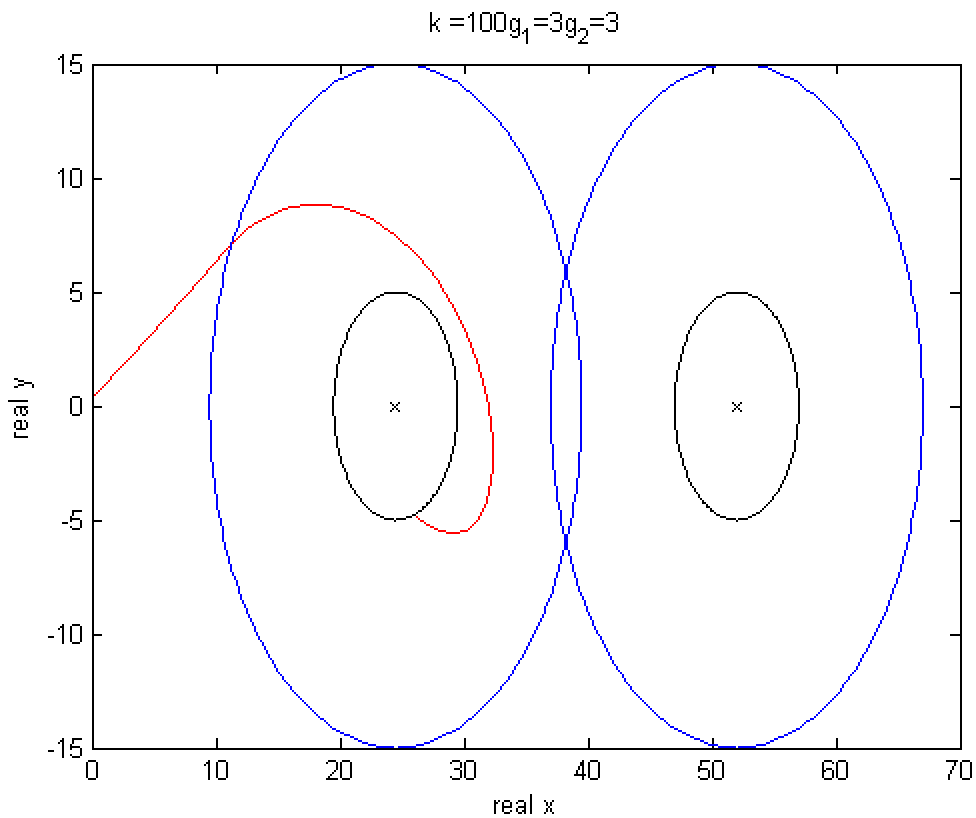

k =50g1=10g2=10
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70

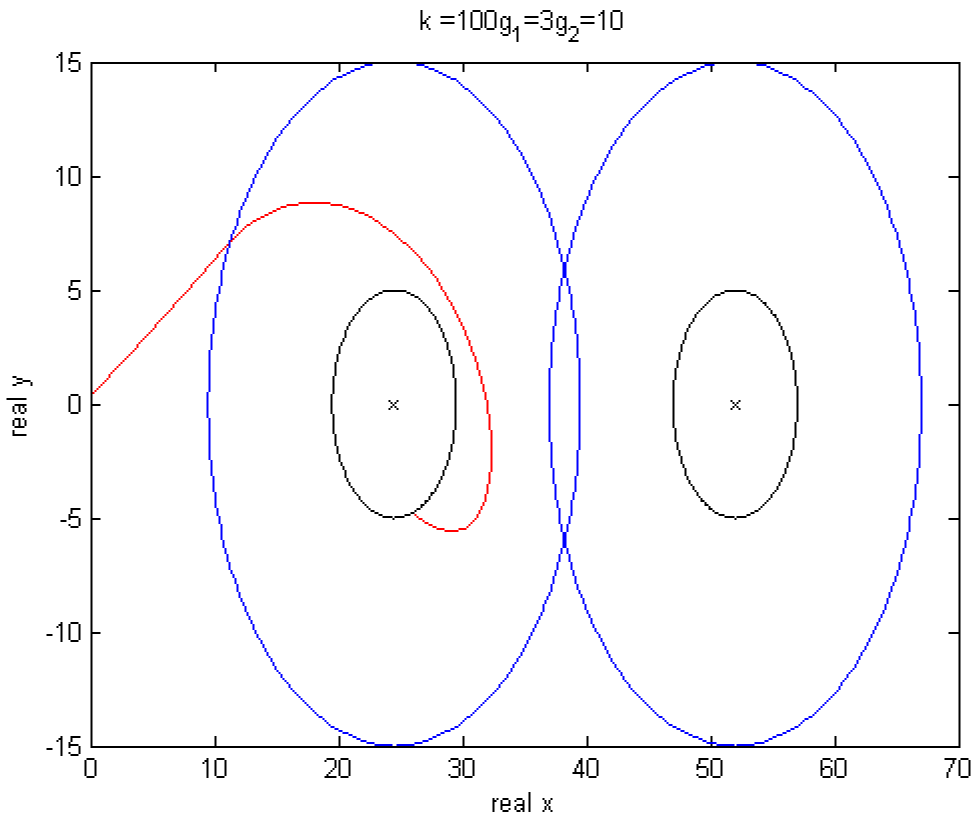

k =100g1=1g2=1
real y
real x
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70

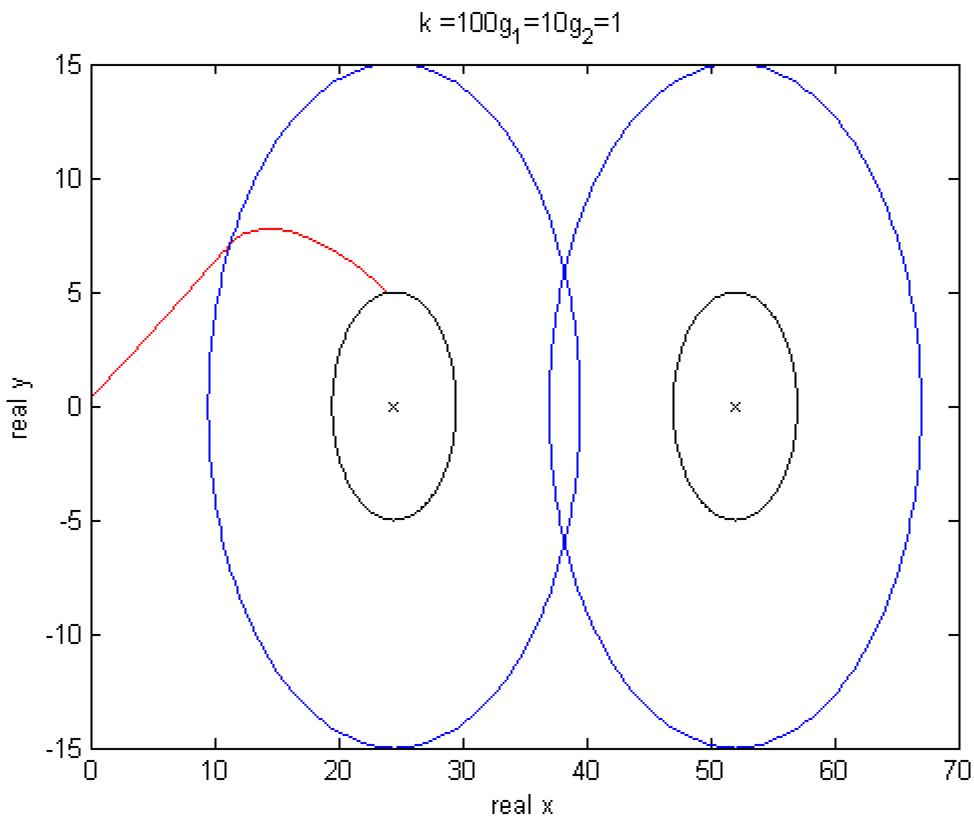

k =100g1=1g2=3
15
10
5
0
-5
-10
-15
real y
0
10
20
30
40
50
60
70
real x

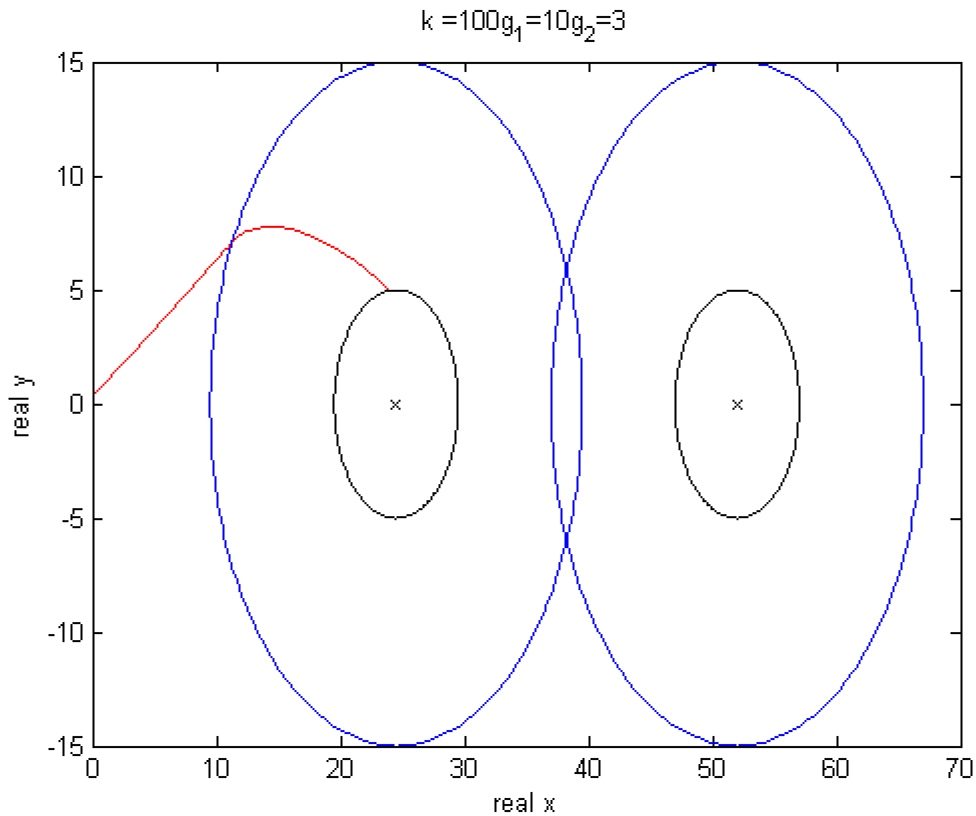

k =100g1=1g2=10
15
10
5
0
-5
-10
-15
real y
0
10
20
30
40
50
60
70
real x

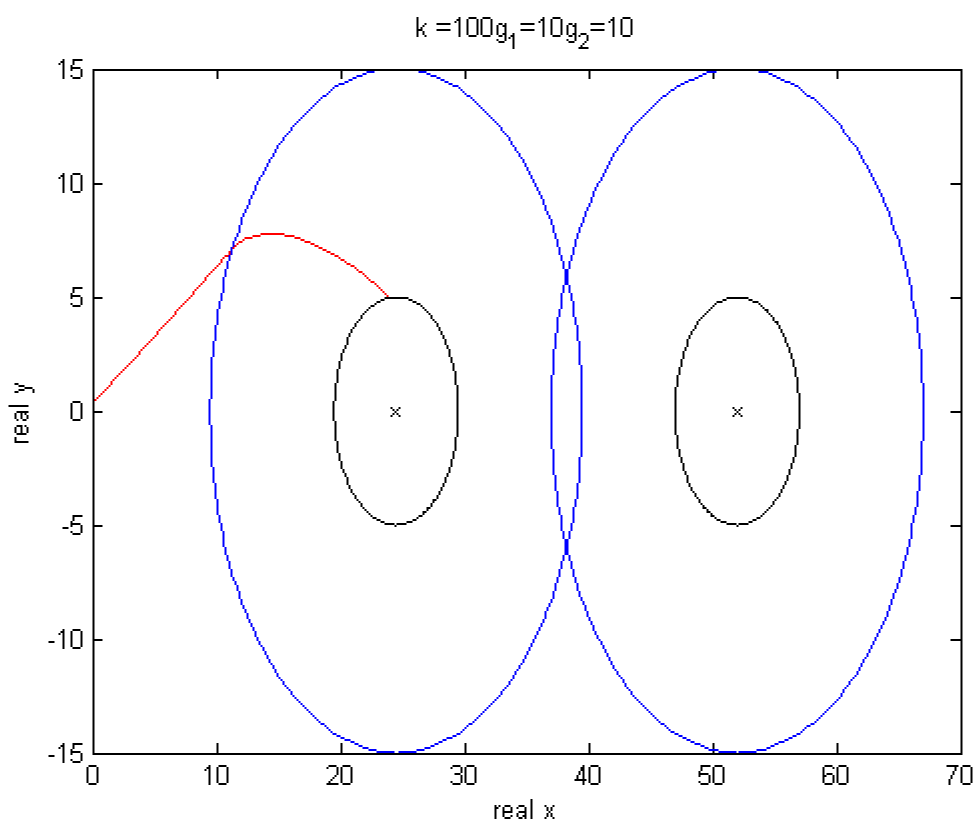

k =100g1=3g2=1
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70
real y
real x


k =100g1=3g2=3
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70
real y
real x


k =100g1=3g2=10
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70
real x
real y


k =100g1=10g2=1
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70
real x
real y


k =100g1=10g2=3
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70
real y
real x


k =100g1=10g2=10
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
60
70
real y
real x

Published with MATLAB® 7.10