Homework4

Problem 4 Solution

findCollision

```
function [t, collisionState] = findCollision (ballState, wall,
coefficient_of_restitution)
    % 先把位置和速度拎出来
    x = ballState(1);
    y = ballState(2);
    vx = ballState(3);
    vy = ballState(4);

    flag = 0;%检测是否有碰撞

    if wall(1) == wall(3) && (wall(1) - x) / vx > 0%竖直
        t = (wall(1) - x) / vx;
        collisionState(1) = wall(1);
        collisionState(2) = y + t * vy;
        %确定是撞到线段上
        if (collisionState(2) - wall(2)) * (collisionState(2) - wall(4)) < 0
            collisionState(3) = - coefficient_of_restitution * vx;
            collisionState(4) = coefficient_of_restitution * vy;
            flag = 1;
        end
    elseif wall(2) == wall(4) && (wall(2) - y) / vy > 0%水平
        t = (wall(2) - y) / vy;
        collisionState(1) = x + t * vx;
        collisionState(2) = wall(2);
        if (collisionState(1) - wall(1)) * (collisionState(1) - wall(3)) < 0
            collisionState(3) = coefficient_of_restitution * vx;
            collisionState(4) = - coefficient_of_restitution * vy;
            flag = 1;
        end
    end

    %没有碰撞
    if flag == 0
        t = Inf;
        collisionState = [];
    end
end
```

updateBallState

```
function newBallState = updateBallState (ballState, dt, walls,
coefficient_of_restitution)
```

```
    Size = size(walls);

    % preallocation
    t = zeros(1,Size(1));
    CollisionState = zeros(1,4);

    for i = 1:Size(1)
        t(i) = findCollision(ballState, walls(i,:), coefficient_of_restitution);
    end

    if dt < min(t)
        newBallState =
[ballState(1)+ballState(3)*dt,ballState(2)+ballState(4)*dt,ballState(3),ballState(
4)];
    else
        i = find(t == min(t));
        [~,CollisionState] = findCollision(ballState, walls(i,:),
coefficient_of_restitution);

        newBallState = [CollisionState(1) + CollisionState(3) * (dt -
min(t)),CollisionState(2) + CollisionState(4) * (dt-
min(t)),CollisionState(3),CollisionState(4)];
    end
end
```

It is noteworthy that the timestep should be small enough, or there will be some bugs.

Problem 2 Solution

msd

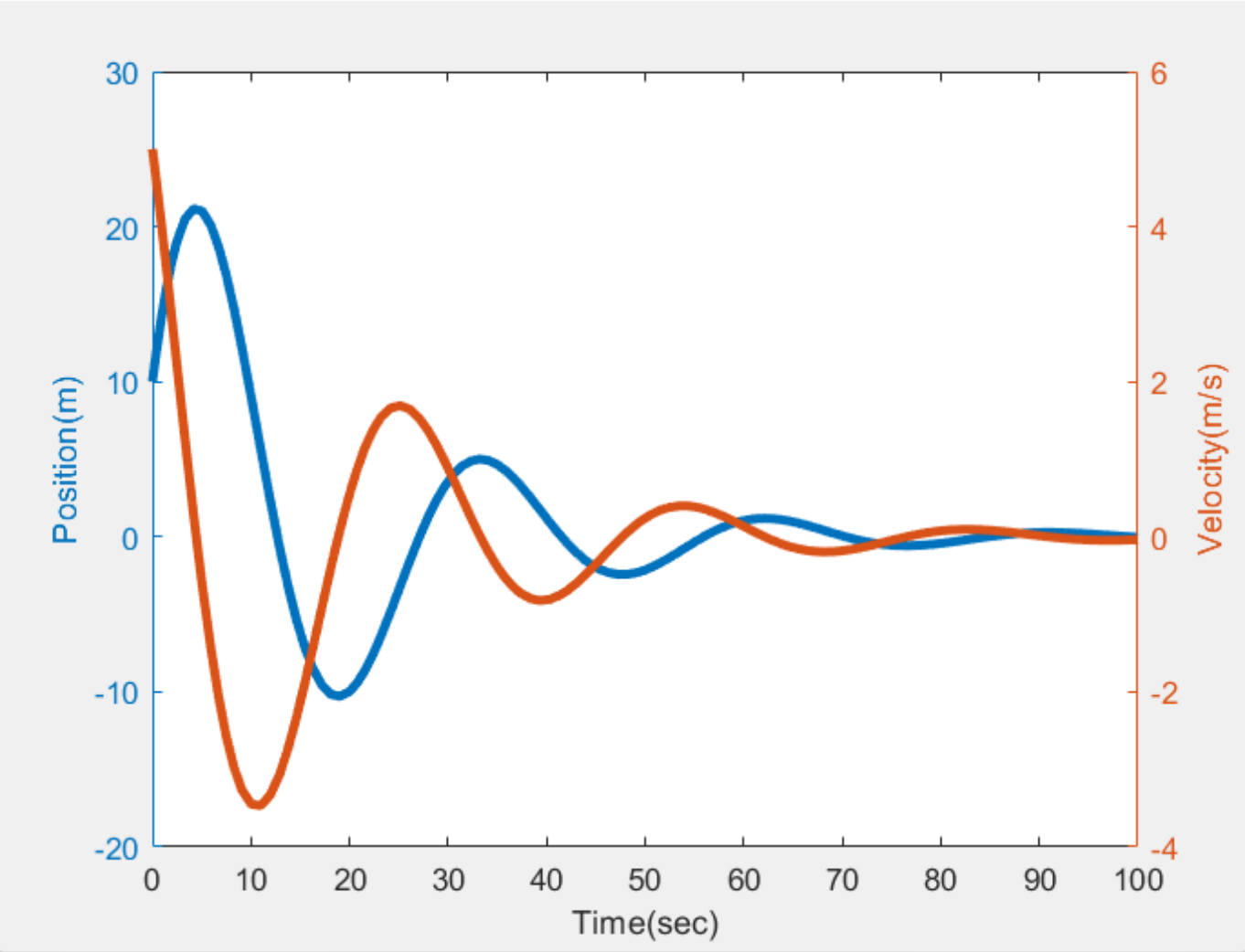
```
function dy = msd(t,y,c,k,m)
    dy = zeros(2,1);
    dy(1) = y(2);
    dy(2) = -(c*y(2)+k*y(1))/m;
end
```

msd_posVel

```
function msd_posVel(c,k,m,yi,vi,t_beg,t_end)
    y0 = [yi,vi];
    tspan = [t_beg,t_end];
    [t,y] = ode45(@msd,tspan,y0,[],c,k,m);

    [hAx,L1,L2] = plotyy(t,y(:,1),t,y(:,2));

    xlabel('Time(sec)');
    ylabel(hAx(1),'Position(m)');
```

```
    ylabel(hAx(2),'Velocity(m/s)');
    set(L1,'LineWidth',3);
    set(L2,'LineWidth',3);
end
```

Got the figure:

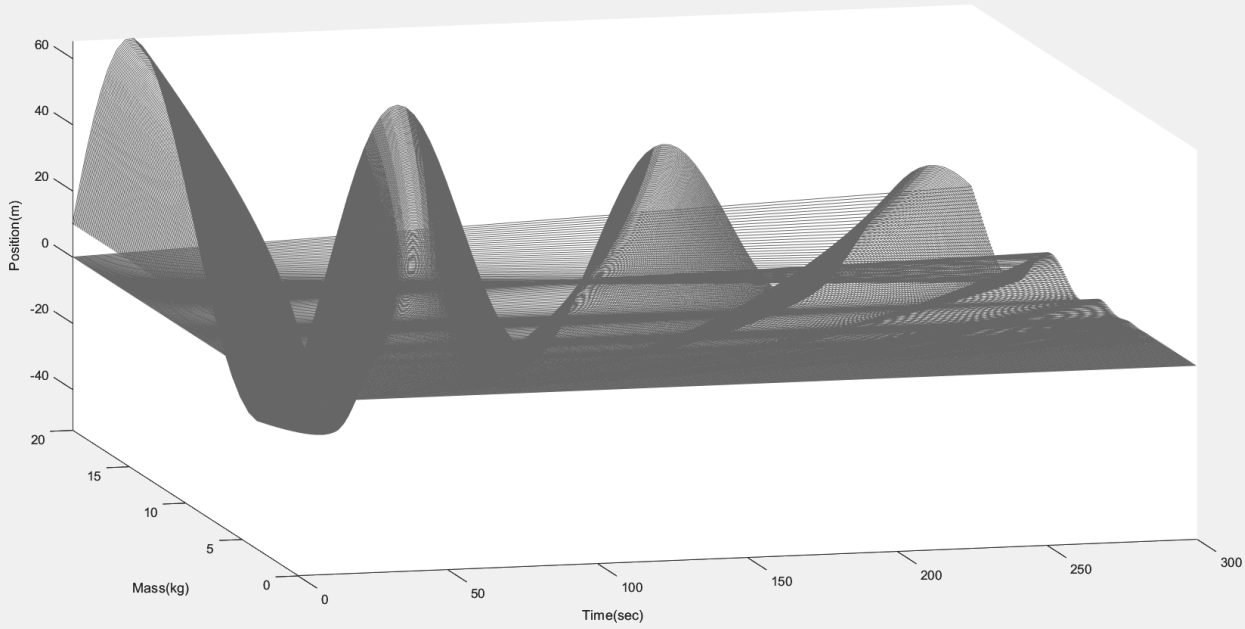


msd_water

```
function msd_water(c,k,m,yi,vi,t_beg,t_end)
    y0 = [yi,vi];
    tspan = [t_beg,t_end];
    [t,y] = ode45(@msd,tspan,y0,[],c,k,min(m));

    time = zeros(length(t),length(m));
    pos = zeros(length(t),length(m));

    mass = repmat(m,length(t),1);
    for i=1:length(m)
        [t,y] = ode45(@msd,tspan,y0,[],c,k,m(i));
        for j=1:length(t)
            time(j,i)=t(j,1);
            pos(j,i)=y(j,1);
```

```
        end
    end
    plot3(time,mass,pos,'Color',[0.4 0.4 0.4]);

    xlabel('Time(sec)');
    ylabel('Mass(kg)');
    zlabel('Position(m)');
end
```

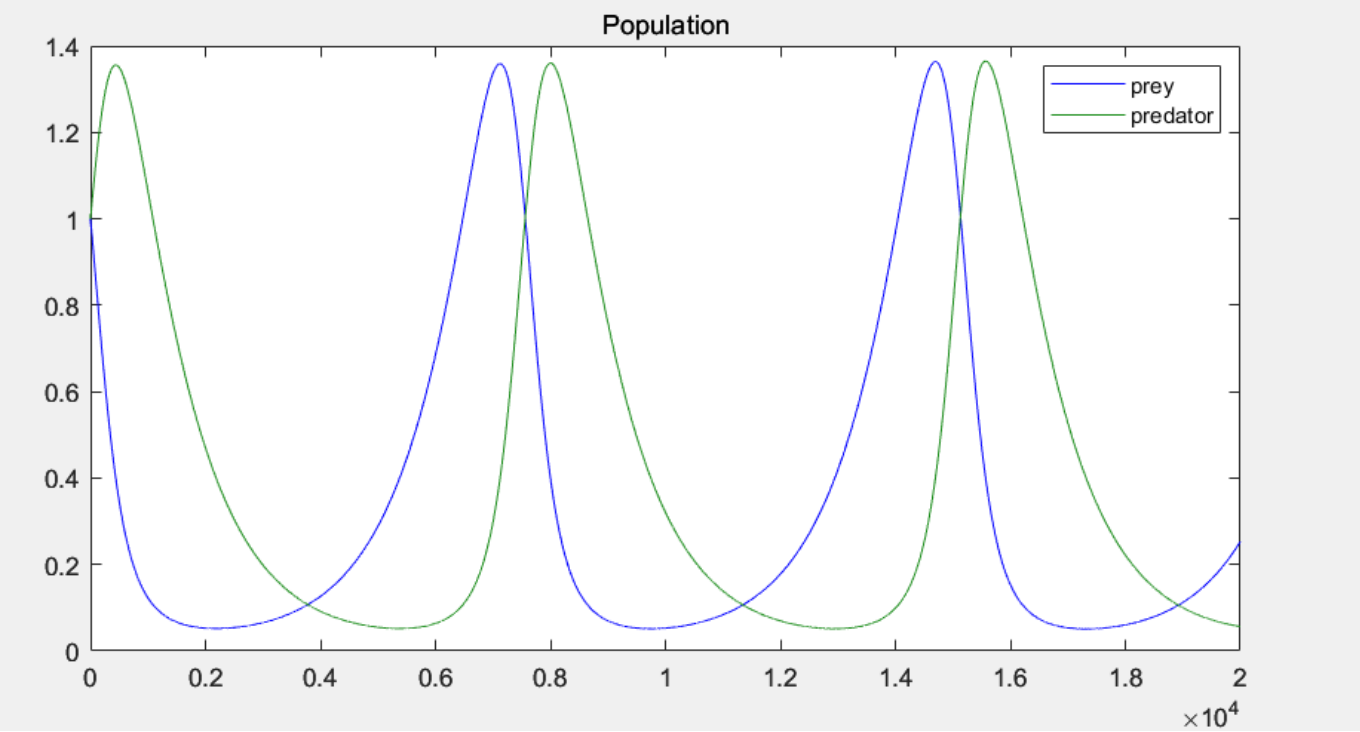
Get the figure:



Problem 3 Solution

```
function state = lotkaVolterra (initial_state, alpha, beta, gamma, delta, dt,
ntimesteps)
    state=zeros(2,ntimesteps);
    state(:,1)=initial_state;

    for i=1:ntimesteps-1
        state(1,i+1)=state(1,i)+dt*state(1,i)*(alpha-beta*state(2,i));
        state(2,i+1)=state(2,i)-dt*state(2,i)*(gamma-delta*state(1,i));
    end

    x = 1 : ntimesteps;
    plot(x,state(1,:),'b');
    hold on
    plot(x,state(2,:),'Color',[27 141 27]/255);
    hold off
    set(gcf,'Position',[0 0 800 400]);
    legend('prey','predator');
end
```

Get the figure:



Problem 4 Solution

do

I make this function to change the distance

```
function out = do(new,distance,occupancy,i,j,count)
    out = new;
    S = size(distance);
    if i>0 && i<=S(1) && j>0 && j<=S(2) && occupancy(i,j) == true && distance(i,j)
== Inf
        out(i,j) = count;
    end
end
```

change

There are four numbers may be changed

```
function new = change(distance,occupancy,i,j,count)
    new = distance;
    new = do(new,distance,occupancy,i+1,j,count);
    new = do(new,distance,occupancy,i-1,j,count);
    new = do(new,distance,occupancy,i,j+1,count);
    new = do(new,distance,occupancy,i,j-1,count);
end
```

grassfire

```
function distance = grassfire (occupancy, dest_row, dest_col)
% initialization
    Size = size(occupancy);
    distance = zeros(Size);
    distance(:,:) = Inf;
    distance(dest_row, dest_col) = 0;
    for count = 0:100
        Temp = distance;
        for i=1:Size(1)
            for j=1:Size(2)
                if distance(i,j) == count
                    distance =
                end
            end
        end

        if Temp == distance
            break;
        end
    end
end
```

We could get the answer:

```
occupancy =

  3×3 logical 数组

   1   1   1
   1   0   0
   1   0   1

>> grassfire(occupancy, 3,3)

ans =

   Inf   Inf   Inf
   Inf   Inf   Inf
   Inf   Inf     0

>>
```

```
occupancy =

  3×3 logical 数组

   1   1   1
   1   1   1
   1   1   1

>> grassfire(occupancy, 3,3)

ans =

     4     3     2
     3     2     1
     2     1     0

>>
```