```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cucker-Smale model
% See F. Cucker, S. Smale, Emergent Behavior in Flocks
% https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4200853
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% xm'(t) = vm(t)
% psilm = phi(|xl(t) - xm(t)|) = 1/(alpha^2 + |xl(t) - xm(t)|^2)^beta
% vm'(t) = C/M sum_{l=1}^{M} psilm (vl(t) - vm(t))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all
clc
format short e
pause off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simulation of flocks
% TestCase = 1
% Numerical Comparisons
% TestCase = 0
% Temporal orders
% TestCase = -1
% Visualisation of results for systems on networks
% TestCase = 100
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

TestCase = 1

% Flock 1/2/3/4
if TestCase == 1
    Param.nDisplayYes = 0;
    Flock = 1
    Flock1234(TestCase,Flock,Param);
    Flock = 2
    Flock1234(TestCase,Flock,Param);
    Flock = 3
    Flock1234(TestCase,Flock,Param);
    Flock = 4
    Flock1234(TestCase,Flock,Param);
end

if TestCase == 0
    for d = 2:3
        clear Param
        close all
        Param.d = d;
        if d == 2
            Param.K0 = 20;
        end
        if d == 3
            Param.K0 = 10;
        end
        Param = SetParametersPlot(TestCase,Param);
        Param.Color1 = Param.ColorMyBlue;
```

```
Param.Color1a = Param.ColorMyBlueLighter2;
Param.Color2 = Param.ColorMyRed;
Param.Color2a = Param.ColorMyRedLighter2;
Param.Color3 = Param.ColorMyGrey;
Param.Color4 = Param.ColorMyGreen;
Param = SetParametersCuckerSmale(Param);
Param.nDisplayYes = 0;
% MM = [100,200,500,1000,2000,5000,10000]';
MM = [100,200,500,1000]';
for loop = 1:length(MM)
    MM(loop)
    Param.M = MM(loop);
    Param = SetParametersSingleStep(Param);
    Param = SetInitialConditionRandom(Param);
    % Time integration by standard approach
    Param.NovelApproachYes = 0;
    [ResultStandard,Param] = TimeIntegration(Param);
    CT_StandardTimeIntegration(loop,1) ...
    = ResultStandard.CT_StandardTimeIntegration;
    % Time integration by novel approach
    Param.NovelApproachYes = 1;
    [ResultNovel,Param] = TimeIntegration(Param);
    CT_NovelApproachPrecomputation(loop,1) ...
    = ResultNovel.CT_NovelApproachPrecomputation;
    CT_NovelApproachTimeIntegration(loop,1) ...
    = ResultNovel.CT_NovelApproachTimeIntegration;
    % Differences
    DifferenceP ...
    = ResultNovel.Position{end} - ResultStandard.Position{end};
    ErrorP(loop,1) = MyMaxAbs(Param.AsColumn(DifferenceP),Param.d);
    DifferenceV ...
    = ResultNovel.Velocity{end} - ResultStandard.Velocity{end};

    ErrorV(loop,1) = MyMaxAbs(Param.AsColumn(DifferenceV),Param.d);
end
Marker1a = 'x-';
Marker1 = 'x-';
Marker2a = 'x-';
Marker2 = 'x-';
Marker4 = 'x-';
loglog(MM,CT_StandardTimeIntegration, ...
Marker1,'Color',Param.Color1, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
hold on
loglog(MM,CT_NovelApproachTimeIntegration, ...
Marker2,'Color',Param.Color2, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
grid on
axis([min(MM),max(MM),1e-3,1e3])
set(gca,'Color',Param.Color3);
set(gcf,'Color',Param.Color3);
set(gca,'FontSize',Param.FontSize);
set(gcf,'InvertHardCopy','off');
title({['Computation times versus number of individuals']; ...
['(',num2str(Param.d),'d, Number of Fourier functions K = ', ...
```

```
num2str(prod(Param.K)),')'];[]});
xlabel('M')
ylabel('CT [seconds]')
legend('Standard approach','Novel approach','Location','northwest')

if Param.d == 2
    saveas(gcf,'StandardNovel_ComputationTimes_2d.jpg');
end
if Param.d == 3
    saveas(gcf,'StandardNovel_ComputationTimes_3d.jpg');
end
%%%%
close all
loglog(MM, ...
CT_StandardTimeIntegration./CT_NovelApproachTimeIntegration, ...
Marker4,'Color',Param.Color4, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
grid on
axis([min(MM),max(MM),1e-1,1e3])
set(gca,'Color',Param.Color3);
set(gcf,'Color',Param.Color3);
set(gca,'FontSize',Param.FontSize);
set(gcf,'InvertHardCopy','off');
title({['Corresponding ratios of computation times']; ...
['(',num2str(Param.d),'d, Number of Fourier functions K = ', ...
num2str(prod(Param.K)),')'];[]});
xlabel('M')
ylabel('CT [seconds]')
if Param.d == 2
    saveas(gcf,'StandardNovel_ComputationTimesRatios_2d.jpg');
end
if Param.d == 3
    saveas(gcf,'StandardNovel_ComputationTimesRatios_3d.jpg');
end
%%%%
close all
loglog(MM,ErrorV, ...
Marker2a,'Color',Param.Color2a, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
hold on
loglog(MM,ErrorP, ...
Marker2,'Color',Param.Color2, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
grid on
% axis([min(MM),max(MM),0.1*min(ErrorP),100*max(ErrorV)])
axis([min(MM),max(MM),1e-13,1.1*1e-5])
xlabel('M')
ylabel('Differences')
set(gca,'Color',Param.Color3);
set(gcf,'Color',Param.Color3);
set(gca,'FontSize',Param.FontSize);
set(gcf,'InvertHardCopy','off');
title({['Corresponding differences of solutions']; ...
['(',num2str(Param.d),'d, K = ',num2str(prod(Param.K)), ...
', Time increment dt = ',num2str(Param.dt),')'];[]});
```

```
        legend('Velocity','Position')
        if Param.d == 2
            saveas(gcf,'StandardNovel_ErrorsSolutions_2d.jpg');
        end
        if Param.d == 3
            saveas(gcf,'StandardNovel_ErrorsSolutions_3d.jpg');
        end
    end
end

if TestCase == -1
    for d = 2:3
        clear Param
        close all
        Param.d = d;
        Param.NovelApproachYes = 0;
        Param = SetParametersPlot(TestCase,Param);
        Param = SetParametersCuckerSmale(Param);
        Param.M = 30;
        Param = SetParametersTimeIntegration(Param);
        Param = SetInitialConditionRandom(Param);
        Param.nDisplayYes = 0;
        % Temporal orders
        if Param.d == 2
            File = 'MyPlot_CuckerSmale_TemporalOrders_2d.jpg';
        end
        if Param.d == 3
            File = 'MyPlot_CuckerSmale_TemporalOrders_3d.jpg';
        end
        AllTemporalOrders(File,Param);
    end
end

if TestCase == 100
    disp('Results for Cucker-Smale and Kuramoto-Daido models')
    disp('(Artificially generated graphs)')
    % load CS_KD_ArtificialGraph.mat
    MM = [100,200,500,1000,2000,5000,10000]';
    Times_CS_Standard ...
    = [9.4015e-02,3.0236e-01,1.5493e+00,8.3001e+00,3.4589e+01, ...
       2.1156e+02,8.3310e+02]';
    Times_KD_Standard ...
    = [2.5015e-02,4.0482e-02,1.9222e-01,9.5456e-01,4.8794e+00, ...
       3.4978e+01,1.2107e+02]';
    Times_CS_Novel ...
    = [1.9083e-01,3.0514e-01,5.8899e-01,1.0572e+00,1.9335e+00, ...
       4.8282e+00,1.0154e+01]';
    Times_KD_Novel ...
    = [4.6938e-02,5.9144e-02,1.2051e-01,2.0429e-01,3.6720e-01, ...
       7.5092e-01,1.3105e+00]';
    x = Times_CS_Standard
    Slope = log(x(1:end-1)./x(2:end))./log(MM(1:end-1)./MM(2:end))
    x = Times_KD_Standard
    Slope = log(x(1:end-1)./x(2:end))./log(MM(1:end-1)./MM(2:end))
    x = Times_CS_Novel
```

```
    Slope = log(x(1:end-1)./x(2:end))./log(MM(1:end-1)./MM(2:end))
    x = Times_KD_Novel
    Slope = log(x(1:end-1)./x(2:end))./log(MM(1:end-1)./MM(2:end))
    Param = [];
    Param = SetParametersPlot(TestCase,Param);
    Param.Color1 = Param.ColorMyBlue;
    Param.Color1a = Param.ColorMyBlueLighter2;
    Param.Color2 = Param.ColorMyRed;
    Param.Color2a = Param.ColorMyRedLighter2;
    Param.Color3 = Param.ColorMyGrey;
    Marker1a = 'x-';
    Marker1 = 'x-';
    Marker2a = 'x-';
    Marker2 = 'x-';
    loglog(MM,Times_CS_Standard, ...
    Marker1a,'Color',Param.Color1a, ...
    'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
    hold on
    loglog(MM,Times_KD_Standard, ...
    Marker1,'Color',Param.Color1, ...
    'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
    loglog(MM,Times_CS_Novel, ...
    Marker2a,'Color',Param.Color2a, ...
    'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
    loglog(MM,Times_KD_Novel, ...
    Marker2,'Color',Param.Color2, ...
    'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
    grid on
    axis([min(MM),max(MM),1e-2,1e4])
    set(gca,'Color',Param.Color3);
    set(gcf,'Color',Param.Color3);
    set(gca,'FontSize',Param.FontSize);
    set(gcf,'InvertHardCopy','off');
    title({['Computation times versus number of components']; ...
    ['Cucker-Smale (CS), Kuramoto-Daido (KD)'];[]});
    xlabel('M')
    ylabel('CT [seconds]')
    legend('Standard approach (CS)','Standard approach (KD)', ...
           'Novel approach (CS)','Novel approach (KD)', ...
           'Location','northwest')
    saveas(gcf,'StandardNovel_ComputationTimes_CSKD.jpg');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Settings
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = SetParametersCuckerSmale(Param)

Param.C = 1;
Param.alpha = 1;
Param.beta = 1;

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = SetParametersSingleStep(Param)

Param.t0 = 0;
Param.N = 1;
Param.dt = 1e-03;
Param.T = Param.t0 + Param.dt;
Param.RKM = 2;
Param.SaveSolutionYes = 1;
Param.AsColumn = inline('reshape(f,prod(size(f)),1)');

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = SetParametersTimeIntegration(Param)

Param.t0 = 0;
% Param.T = 15;
Param.T = 1;
Param.N = 20*Param.T;
Param.RKM = 2;
Param.SaveSolutionYes = 1;
Param.AsColumn = inline('reshape(f,prod(size(f)),1)');

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = SetInitialConditionRandom(Param)

Param.Position0 = rand(Param.d,Param.M);
Param.Velocity0 = 1/10*rand(Param.d,Param.M);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = SetInitialCondition(Param)

if Param.d == 1
    Direction = - 1;
end
if Param.d == 2
    Direction = [-1,1]';
end
if Param.d == 3
    Direction = [-1,1,0]';
end
if Param.IC == 'L'
    DirectionVelocity = - Direction;
end
if Param.IC == 'T'
    if Param.d == 1
```

```
        DirectionVelocity = Direction;
    end
    if Param.d == 2
        Direction = [-1,1]';
        DirectionVelocity = [1,0];
    end
    if Param.d == 3
        DirectionVelocity = [1,0,0];
    end
end
if Param.IC == 'S'
    DirectionVelocity = Param.DirectionVelocity;
end
for m = 1:Param.M
    Velocity(:,m) = DirectionVelocity;
end

% Line
if Param.IC == 'L'
    for m = 1:Param.M
        Position(:,m) = (m-1)/(Param.M-1)*Direction;
    end
end

% Tringle
if Param.IC == 'T'
    if Param.d == 1
        DirectionOrthogonal = 1;
    end
    if Param.d == 2
        DirectionOrthogonal = -[1,1]';
    end
    if Param.d == 3
        DirectionOrthogonal = [1,1,0]';
    end
    for m = 1:Param.M1
        Position(:,m) = (m-1)/(Param.M1-1)*Direction;
    end
    for m = Param.M1+1:Param.M
        Position(:,m) = (m-Param.M1)/(Param.M2-1)*DirectionOrthogonal;
    end
end

% Strip M = MFactor1 x MFactor2
if Param.IC == 'S'
    if Param.d == 1
        Param.MFactor1 = Param.M;
        Param.MFactor2 = 1;
    end
    for loop = 1:Param.MFactor2
        IndMin = (loop-1)*Param.MFactor1 + 1;
        IndMax = loop*Param.MFactor1;
        Shift = zeros(Param.d,1);
        if Param.d > 1
            Shift(2) = loop-1;
```

```
        end
        for m = IndMin:IndMax
            Factor = (m-(loop-1)*Param.MFactor1-1)/(Param.MFactor1-1);
            Position(:,m) = Shift + Factor*Direction;
        end
    end
end

InitialPointPosition = 10;
InitialPointVelocity = 0;
ScalingPosition = 5;
ScalingVelocity = 1;
if Param.RandomPerturbationYes == 0
    Param.ScalingRandomPerturbationPosition = 0;
    Param.ScalingRandomPerturbationVelocity = 0;
end
Position = InitialPointPosition + ScalingPosition*Position;
Velocity = InitialPointVelocity + ScalingVelocity*Velocity;
if Param.RandomPerturbationYes == 1
    for i = 1:Param.d
        Position(i,:) ...
        = Position(i,:) ...
        + Param.ScalingRandomPerturbationPosition(i) ...
        *rand(size(Position(i,:)));
        Velocity(i,:) ...
        = Velocity(i,:) ...
        + Param.ScalingRandomPerturbationVelocity(i) ...
        *rand(size(Velocity(i,:)));
    end
end
Param.Position0 = Position;
Param.Velocity0 = Velocity;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = SetParametersTemporalOrder(Param)

Param.t0 = 1;
Param.T = 2;
Param.N0 = 10;
Param.Factor = 2;
Param.LoopMin = 1;
% Param.LoopMax = 10;
Param.LoopMax = 5;
Param.SaveSolutionYes = 0;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = SetParametersPlot(TestCase,Param)

Param.FontSize = 20;
```

```
Param.Marker = '.';
Param.MarkerFirst = '.';
Param.MarkerLast = '.';
if TestCase == 1
    Param.MarkerSize = 10;
end
if TestCase == -1 | TestCase == 0 | TestCase == 100
    Param.MarkerSize = 12;
end
Param.LineWidth = 2;
Param.LineWidth = 4;
Param.nMod = 2;
Param.ColorLightBlue = [0.7,0.8,0.95];
Param.ColorWhite = [1,1,1];
Param.ColorBlack = [0,0,0];
% d36e70
Param.ColorMyRed = [82.75,43.14,43.92]/100;
% 6a93b0
Param.ColorMyBlue = [41.57,57.65,69.02]/100;
% 7fb0b2
Param.ColorMyGreen = [49.8,69.02,69.8]/100;
% dd8686
Param.ColorMyRedLighter1 = [86.67,52.55,52.55]/100;
% 6daad4
Param.ColorMyBlueLighter1 = [42.75,66.67,83.14]/100;
% e69e9d
Param.ColorMyRedLighter2 = [90.2,61.96,61.57]/100;
% 83a4bd
Param.ColorMyBlueLighter2 = [51.37,64.31,74.12]/100;
% dbdbdb
Param.ColorMyGrey = [85.88,85.88,85.88]/100;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cucker-Smale model
% Defining right-hand side (d = 3)
% xm1'(t) = vm1(t)
% xm2'(t) = vm2(t)
% xm3'(t) = vm3(t)
% psilm = phi(|xl(t) - xm(t)|) = 1/(alpha^2 + |xl(t) - xm(t)|^2)^beta
% vm1'(t) = C/M sum_{l=1}^{M} psiml (vl1(t) - vm1(t))
% vm2'(t) = C/M sum_{l=1}^{M} psiml (vl2(t) - vm2(t))
% vm3'(t) = C/M sum_{l=1}^{M} psiml (vl3(t) - vm3(t))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Result = CuckerSmaleRHS(xv,Param)

Position0 = xv([1:Param.d],:);
Velocity0 = xv([Param.d+1:2*Param.d],:);
% xm1'(t) = vm1(t)
% xm2'(t) = vm2(t)
% xm3'(t) = vm3(t)
Position = Velocity0;
% vm1'(t) = C/M sum_{l=1}^{M} psiml (vl1(t) - vm1(t))
```

```
% vm2'(t) = C/M sum_{l=1}^{M} psiml (vl2(t) - vm2(t))
% vm3'(t) = C/M sum_{l=1}^{M} psiml (vl3(t) - vm3(t))
Velocity = zeros(Param.d,Param.M);
for m = 1:Param.M
    Velocity(:,m) = 0;
    for l = 1:Param.M
        % psilm = phi(|xl(t) - xm(t)|)
        % = 1/(alpha^2 + |xl(t) - xm(t)|^2)^beta
        psiml ...
        = 1./(Param.alpha^2 ...
          + norm(Position0(:,l)-Position0(:,m))^2).^Param.beta;
        Velocity(:,m) ...
        = Velocity(:,m) + psiml*(Velocity0(:,l)-Velocity0(:,m));
    end
    Velocity(:,m) = Param.C/Param.M*Velocity(:,m);
end
Result = [Position;Velocity];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function value = MySum(s,d)

if d == 1
    value = sum(s);
end
if d == 2
    value = sum(sum(s));
end
if d == 3
    value = sum(sum(sum(s)));
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Maximum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function value = MyMaxAbs(s,d)

s = abs(s);
if d == 1
    value = max(s);
end
if d == 2
    value = max(max(s));
end
if d == 3
    value = max(max(max(s)));
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Novel approach - Norm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function value = NormSquare(xi,Param)

if Param.d == 1
    value = xi{1}.^2;
end
if Param.d == 2
    value = xi{1}.^2 + xi{2}.^2;
end
if Param.d == 3
    value = xi{1}.^2 + xi{2}.^2 + xi{3}.^2;
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Novel approach - Fourier functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function value = E1D(xi,k,a)

value = exp(2*pi*1i*k/(a(2)-a(1))*xi);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function value = Fourier1D(xi,k,a)

value = 1/sqrt(a(2)-a(1))*exp(2*pi*1i*k/(a(2)-a(1))*(xi+a(1)));

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function value = Fourier(xi,k,a)

value = Fourier1D(xi{1},k(1),a{1});
for i = 2:length(xi)
    value = value.*Fourier1D(xi{i},k(i),a{i});
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Novel approach - Fourier coefficients (Real2Spectral)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function phis = FourierCoefficients(phi,Param)
```

```
if Param.ByFFTYes == 0
    phis = zeros(size(phi));
    dxi = prod(Param.dxi);
    Shift = Param.K/2+1;
    if Param.d == 1
        for k = -Param.K/2:Param.K/2-1
            phis(k+Shift,1) ...
            = dxi*MySum(phi.*Fourier(Param.xi,-k,Param.a),Param.d);
        end
    end
    if Param.d == 2
        for k1 = -Param.K(1)/2:Param.K(1)/2-1
            for k2 = -Param.K(2)/2:Param.K(2)/2-1
                k = [k1,k2];
                phis(k1+Shift(1),k2+Shift(2)) ...
                = dxi*MySum(phi.*Fourier(Param.xi,-k,Param.a),Param.d);
            end
        end
    end
    if Param.d == 3
        for k1 = -Param.K(1)/2:Param.K(1)/2-1
            for k2 = -Param.K(2)/2:Param.K(2)/2-1
                for k3 = -Param.K(3)/2:Param.K(3)/2-1
                    k = [k1,k2,k3];
                    phis(k1+Shift(1),k2+Shift(2),k3+Shift(3)) ...
                    = dxi*MySum(phi.*Fourier(Param.xi,-k,Param.a),Param.d);
                end
            end
        end
    end
end

if Param.ByFFTYes == 1
    Constr2s = 1/prod(Param.K);
    for i = 1:Param.d
        Constr2s = sqrt(Param.a{i}(2)-Param.a{i}(1))*Constr2s;
    end
    phis = Constr2s*fftshift(fftn(phi));
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Novel approach - Fourier series (Spectral2Real)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function value = FourierSeries(xi,phis,Param)

if Param.ByFFTYes == 0
    Shift = Param.K/2+1;
    if Param.d == 1
        value = 0;
        for k = -Param.K/2:Param.K/2-1
            value = value + phis(k+Shift)*Fourier(xi,k,Param.a);
        end
```

```
    end
    if Param.d == 2
        value = 0;
        for k1 = -Param.K(1)/2:Param.K(1)/2-1
            for k2 = -Param.K(2)/2:Param.K(2)/2-1
                k = [k1,k2];
                value = value ...
                + phis(k1+Shift(1),k2+Shift(2))*Fourier(xi,k,Param.a);
            end
        end
    end
    if Param.d == 3
        value = 0;
        for k1 = -Param.K(1)/2:Param.K(1)/2-1
            for k2 = -Param.K(2)/2:Param.K(2)/2-1
                for k3 = -Param.K(3)/2:Param.K(3)/2-1
                    k = [k1,k2,k3];
                    value = value ...
                    + phis(k1+Shift(1),k2+Shift(2),k3+Shift(3)) ...
                    *Fourier(xi,k,Param.a);
                end
            end
        end
    end
end

if Param.ByFFTYes == 1
    Consts2r = prod(Param.K);
    for i = 1:Param.d
        Consts2r = 1/sqrt(Param.a{i}(2)-Param.a{i}(1))*Consts2r;
    end
    value = Consts2r*ifftn(ifftshift(phis));
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cucker-Smale model
% xm'(t) = vm(t)
% psilm = phi(|xl(t) - xm(t)|) = 1/(alpha^2 + |xl(t) - xm(t)|^2)^beta
% vm'(t) = C/M sum_{l=1}^{M} psiml (vl(t) - vm(t))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Novel approach for fast summation based on Fourier approximation
% C/M sum_{k1,k2,k3} phisk1k2k3 Fk1(xm1(t)) Fk2(xm2(t)) Fk3(xm3(t))
% (sum_{l=1}^{M} Ek1(-xl1(t)) Ek2(-xl2(t)) Ek2(-xl3(t)) vl(t)
% - vm(t) sum_{l=1}^{M} Ek1(-xl1(t)) Ek2(-xl2(t)) Ek3(-xl3(t))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Result = CuckerSmaleRHSNovelApproach(xv,Param)

Position0 = xv([1:Param.d],:);
Velocity0 = xv([Param.d+1:2*Param.d],:);

Position = Velocity0;
```

```
Shift = Param.K/2+1;
if Param.d == 1
    Value1 = 0;
    Value2 = 0;
    for k = -Param.K/2:Param.K/2-1
        Aux = E1D(-Position0,k,Param.a{1});
        S1 = sum(Aux.*Velocity0);
        S2 = sum(Aux);
        Aux = Param.phis(k+Shift)*Fourier1D(Position0,k,Param.a{1});
        Value1 = Value1 + S1*Aux;
        Value2 = Value2 + S2*Aux;
    end
    Velocity = Param.C/Param.M*(Value1 - Velocity0.*Value2);
end
if Param.d == 2
    Value11 = 0;
    Value12 = 0;
    Value2 = 0;
    for k1 = -Param.K(1)/2:Param.K(1)/2-1
        for k2 = -Param.K(2)/2:Param.K(2)/2-1
            Aux = E1D(-Position0(1,:),k1,Param.a{1}) ...
                .*E1D(-Position0(2,:),k2,Param.a{2});
            S11 = sum(Aux.*Velocity0(1,:));
            S12 = sum(Aux.*Velocity0(2,:));
            S2 = sum(Aux);
            Aux = Param.phis(k1+Shift(1),k2+Shift(2)) ...
                 *Fourier1D(Position0(1,:),k1,Param.a{1}) ...
                .*Fourier1D(Position0(2,:),k2,Param.a{2});
            Value11 = Value11 + Aux*S11;
            Value12 = Value12 + Aux*S12;
            Value2 = Value2 + Aux*S2;
        end
    end
    Velocity(1,:) = Param.C/Param.M*(Value11 - Velocity0(1,:).*Value2);
    Velocity(2,:) = Param.C/Param.M*(Value12 - Velocity0(2,:).*Value2);
end
if Param.d == 3
    Value11 = 0;
    Value12 = 0;
    Value13 = 0;
    Value2 = 0;
    for k1 = -Param.K(1)/2:Param.K(1)/2-1
        for k2 = -Param.K(2)/2:Param.K(2)/2-1
            for k3 = -Param.K(3)/2:Param.K(3)/2-1
                Aux = E1D(-Position0(1,:),k1,Param.a{1}) ...
                    .*E1D(-Position0(2,:),k2,Param.a{2}) ...
                    .*E1D(-Position0(3,:),k3,Param.a{3});
                S11 = sum(Aux.*Velocity0(1,:));
                S12 = sum(Aux.*Velocity0(2,:));
                S13 = sum(Aux.*Velocity0(3,:));
                S2 = sum(Aux);
                Aux = Param.phis(k1+Shift(1),k2+Shift(2),k3+Shift(3)) ...
                     *Fourier1D(Position0(1,:),k1,Param.a{1}) ...
                    .*Fourier1D(Position0(2,:),k2,Param.a{2}) ...
                    .*Fourier1D(Position0(3,:),k3,Param.a{3});
```

```
                Value11 = Value11 + Aux*S11;
                Value12 = Value12 + Aux*S12;
                Value13 = Value13 + Aux*S13;
                Value2 = Value2 + Aux*S2;
            end
        end
    end
    Velocity(1,:) = Param.C/Param.M*(Value11 - Velocity0(1,:).*Value2);
    Velocity(2,:) = Param.C/Param.M*(Value12 - Velocity0(2,:).*Value2);
    Velocity(3,:) = Param.C/Param.M*(Value13 - Velocity0(3,:).*Value2);
end
Velocity = real(Velocity);
Result = [Position;Velocity];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Time integration of Cucker-Smale model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Result,Param] = TimeIntegration(Param)

% Precomputations (Approximation by Fourier series)
if Param.NovelApproachYes == 1
    tic
    % Grid points
    Param.K = Param.K0*ones(Param.d,1);
    for i = 1:Param.d
        if mod(Param.K(i),2) > 0
            error('Components of K should be even numbers!')
        end
    end
    for i = 1:Param.d
        Param.a{i} = [-2,2];
        a(i,:) = Param.a{i};
        xi{i} = linspace(Param.a{i}(1),Param.a{i}(2),Param.K(i)+1)';
        xi{i} = xi{i}(1:end-1);
        Param.dxi(i,1) = xi{i}(2)-xi{i}(1);
    end
    if Param.d == 1
        Param.xi{1} = xi{1};
    end
    if Param.d == 2
        [Param.xi{1},Param.xi{2}] = ndgrid(xi{1},xi{2});
    end
    if Param.d == 3
        [Param.xi{1},Param.xi{2},Param.xi{3}] ...
        = ndgrid(xi{1},xi{2},xi{3});
    end
    % Defining kernel phi(xi) = 1/(alpha^2 + |xi|^2)^beta
    phi = 1./(Param.alpha^2 + NormSquare(Param.xi,Param)).^Param.beta;
    % Fourier coefficients (use of Fast Fourier techniques)
    Param.ByFFTYes = 1;
    Param.phis = FourierCoefficients(phi,Param);
    Difference = FourierSeries(Param.xi,Param.phis,Param) - phi;
```

```
    M_ErrorApproximationKernelByFourier ...
    = [Param.M,MyMaxAbs(Difference,Param.d)]
    Result.CT_NovelApproachPrecomputation = toc;
end

tic
Param.dt = (Param.T-Param.t0)/Param.N;
xv = [Param.Position0;Param.Velocity0];
if Param.SaveSolutionYes == 1
    Result.Time = Param.t0 + Param.dt*[0:Param.N]';
    Result.Position{1} = Param.Position0;
    Result.Velocity{1} = Param.Velocity0;
end
for n = 1:Param.N
    if Param.nDisplayYes == 1
        n
    end
    % Explicit Euler method
    if Param.RKM == 1
        if Param.NovelApproachYes == 0
            xv = xv + Param.dt*CuckerSmaleRHS(xv,Param);
        end
        if Param.NovelApproachYes == 1
            xv = xv + Param.dt*CuckerSmaleRHSNovelApproach(xv,Param);
        end
    end
    % Explicit midpoint rule
    if Param.RKM == 2
        if Param.NovelApproachYes == 0
            xvAux = xv + Param.dt/2*CuckerSmaleRHS(xv,Param);
            xv = xv + Param.dt*CuckerSmaleRHS(xvAux,Param);
        end
        if Param.NovelApproachYes == 1
            xvAux = xv + Param.dt/2*CuckerSmaleRHSNovelApproach(xv,Param);
            xv = xv + Param.dt*CuckerSmaleRHSNovelApproach(xvAux,Param);
        end
    end
    if Param.SaveSolutionYes == 1
        Result.Position{n+1} = xv(1:Param.d,:);
        Result.Velocity{n+1} = xv(Param.d+1:end,:);
    end
end
if Param.SaveSolutionYes == 0
    Result.Time = Param.t0 + Param.dt*Param.N;
    Result.Position = xv(1:Param.d,:);
    Result.Velocity = xv(Param.d+1:end,:);
end
if Param.NovelApproachYes == 0
    Result.CT_StandardTimeIntegration = toc;
end
if Param.NovelApproachYes == 1
    Result.CT_NovelApproachTimeIntegration = toc;
end

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Temporal order
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ErrorOrder = TemporalOrder(Param)

for loop = Param.LoopMax:-1:Param.LoopMin
    Param.N = round((Param.T-Param.t0)*Param.N0*Param.Factor^loop);
    N(loop,1) = Param.N;
    [Result,Param] = TimeIntegration(Param);
    Position{loop} = Result.Position;
    Velocity{loop} = Result.Velocity;
end
for loop = Param.LoopMin:Param.LoopMax-1
    Error.Position(loop,1) = norm(Position{loop}-Position{Param.LoopMax});
    Error.Velocity(loop,1) = norm(Velocity{loop}-Velocity{Param.LoopMax});
end
Order.Position ...
= - log(Error.Position(2:end)./Error.Position(1:end-1))/log(Param.Factor);
Order.Velocity ...
= - log(Error.Velocity(2:end)./Error.Velocity(1:end-1))/log(Param.Factor);
ErrorOrder.N = N;
ErrorOrder.ErrorPosition = Error.Position;
ErrorOrder.ErrorVelocity = Error.Velocity;
ErrorOrder.NErrorOrderPosition ...
= [N,[0;Error.Position],[0;0;Order.Position]];
ErrorOrder.NErrorOrderVelocity ...
= [N,[0;Error.Velocity],[0;0;Order.Velocity]];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function AllTemporalOrders(File,Param)

Param = SetParametersTemporalOrder(Param);
Param.RKM = 1;
ErrorOrderRKM1 = TemporalOrder(Param);
Param.RKM = 2;
ErrorOrderRKM2 = TemporalOrder(Param);
N_ErrorPosition_Order_RKM1 = ErrorOrderRKM1.NErrorOrderPosition
N_ErrorVelocity_Order_RKM1 = ErrorOrderRKM1.NErrorOrderVelocity
pause
N_ErrorPosition_Order_RKM2 = ErrorOrderRKM2.NErrorOrderPosition
N_ErrorVelocity_Order_RKM2 = ErrorOrderRKM2.NErrorOrderVelocity
pause

Param.Color1 = Param.ColorMyBlue;
Param.Color1a = Param.ColorMyBlueLighter2;
Param.Color2 = Param.ColorMyRed;
Param.Color2a = Param.ColorMyRedLighter2;
Param.Color3 = Param.ColorMyGrey;
Marker1a = 'x-';
Marker1 = 'x-';
```

```
Marker2a = 'x-';
Marker2 = 'x-';
loglog(ErrorOrderRKM1.N(2:end),ErrorOrderRKM1.ErrorPosition, ...
Marker1,'Color',Param.Color1, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
hold on
loglog(ErrorOrderRKM1.N(2:end),ErrorOrderRKM1.ErrorVelocity, ...
Marker1a,'Color',Param.Color1a, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
loglog(ErrorOrderRKM2.N(2:end),ErrorOrderRKM2.ErrorPosition, ...
Marker2,'Color',Param.Color2, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
loglog(ErrorOrderRKM2.N(2:end),ErrorOrderRKM2.ErrorVelocity, ...
Marker2a,'Color',Param.Color2a, ...
'MarkerSize',Param.MarkerSize,'LineWidth',Param.LineWidth)
grid on
axis([min(ErrorOrderRKM1.N(2:end)),max(ErrorOrderRKM1.N(2:end)),1e-11,100])
set(gca,'Color',Param.Color3);
set(gcf,'Color',Param.Color3);
set(gca,'FontSize',Param.FontSize);
set(gcf,'InvertHardCopy','off');
if Param.d == 2
    title({['Temporal orders']; ...
    ['System in 2d with M = ',num2str(Param.M),' individuals'];[]});
end
if Param.d == 3
    title({['Temporal orders']; ...
    ['System in 3d with M = ',num2str(Param.M),' individuals'];[]});
end
xlabel('Number of time steps')
ylabel('Error')
legend('Order 1, Error position','Order 1, Error velocity', ...
'Order 2, Error position','Order 2, Error velocity')
saveas(gcf,File);
pause

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot positions/velocities of individuals
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = MinMaxPositionVelocity(Choice,Result,Param)

if Choice == 'P'
    Solution = Result.Position;
end
if Choice == 'V'
    Solution = Result.Velocity;
end

for i = 1:Param.d
    for n = 0:Param.N
        Min{i}(n+1,1) = min(Solution{n+1}(i,:));
        Max{i}(n+1,1) = max(Solution{n+1}(i,:));
```

```
    end
    Min{i} = min(Min{i});
    Max{i} = max(Max{i});
    if Min{i} == Max{i}
        if Min{i} == 0
            Min{i} = - 1e-16;
            Max{i} = 1e-16;
        end
        if abs(Min{i}) > 0
            Min{i} = Min{i} - 0.1*abs(Min{i});
            Max{i} = Max{i} + 0.1*abs(Max{i});
        end
    end
end

if Choice == 'P'
    Param.MinP = Min;
    Param.MaxP = Max;
end
if Choice == 'V'
    Param.MinV = Min;
    Param.MaxV = Max;
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PlotPositionVelocity(Choice,File,nRange,Result,Param)

close all

if Param.NovelApproachYes == 0
    Title2 = ['System with M = ',num2str(Param.M),' individuals'];
end
if Param.NovelApproachYes == 1
    Title2 = ['System with M = ',num2str(Param.M), ...
              ' individuals, K = ',num2str(prod(Param.K))];
end

if Choice == 'P'
    Solution = Result.Position;
    Min = Param.MinP;
    Max = Param.MaxP;
    Title1 = 'Position at time ';
end
if Choice == 'V'
    Solution = Result.Velocity;
    Min = Param.MinV;
    Max = Param.MaxV;
    Title1 = 'Velocity at time ';
end

for n = nRange
    MySolution1_mn(:,n+1) = Solution{n+1}(1,:);
```

```
    if Param.d > 1
        MySolution2_mn(:,n+1) = Solution{n+1}(2,:);
    end
    if Param.d > 2
        MySolution3_mn(:,n+1) = Solution{n+1}(3,:);
    end
end

MySize = 5;
MyColors = linspace(0,1,length(MySolution1_mn(1,:)));
s = (Param.N-nRange')/Param.N;

for i = 1:3
    MyColorMap(:,i) ...
    = s*Param.ColorWhite(i) + (1-s)*Param.ColorBlack(i);
end
colormap(MyColorMap)

for m = 1:Param.M
    if Param.d == 1
        scatter(MySolution1_mn(m,:),0*MySolution1_mn(m,:), ...
        MySize,MyColors,'filled')
        axis([Min{1},Max{1},-1,1])
        xlabel('1st coordinate')
    end
    if Param.d == 2
        scatter(MySolution1_mn(m,:),MySolution2_mn(m,:), ...
        MySize,MyColors,'filled')
        axis([Min{1},Max{1},Min{2},Max{2}])
        xlabel('1st coordinate')
        ylabel('2nd coordinate')
    end
    if Param.d == 3
        scatter3(MySolution1_mn(m,:),MySolution2_mn(m,:), ...
                 MySolution3_mn(m,:),MySize,MyColors,'filled')
        axis([Min{1},Max{1},Min{2},Max{2},Min{3},Max{3}])
        xlabel('1st coordinate')
        ylabel('2nd coordinate')
        zlabel('3rd coordinate')
    end
    hold on
    grid on
    set(gca,'Color',Param.ColorLightBlue);
    set(gcf,'Color',Param.ColorLightBlue);
    set(gcf,'InvertHardCopy','off');
    set(gca,'FontSize',Param.FontSize);
    title({[Title1,num2str(Result.Time(n+1))];Title2;[]});
    drawnow
    pause
end

if Param.SavePlotYes == 1
    saveas(gcf,File);
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function MoviePositionVelocity(Choice,File,nRange,Result,Param)

close all

if Param.MovieYes == 1
    Color = Param.ColorBlack;
    vidObj = VideoWriter(File);
    open(vidObj);
end

if Param.NovelApproachYes == 0
    Title2 = ['System with M = ',num2str(Param.M),' individuals'];
end
if Param.NovelApproachYes == 1
    Title2 = ['System with M = ',num2str(Param.M), ...
              ' individuals, K = ',num2str(prod(Param.K))];
end

if Choice == 'P'
    Solution = Result.Position;
    Min = Param.MinP;
    Max = Param.MaxP;
    Title1 = 'Position at time ';
end
if Choice == 'V'
    Solution = Result.Velocity;
    Min = Param.MinV;
    Max = Param.MaxV;
    Title1 = 'Velocity at time ';
end

for n = nRange
    if Param.nDisplayYes == 1
        n
    end
    if n == 0
        Marker = Param.MarkerFirst;
        nMod = 1;
    end
    if n > 0 & n < Param.N
        Marker = Param.Marker;
        if Param.MovieYes == 0
            nMod = Param.nMod;
        end
        if Param.MovieYes == 1
            nMod = 1;
        end
    end
    if n == Param.N
        Marker = Param.MarkerLast;
        nMod = 1;
```

```
    end
    if n == 0 | mod(n,nMod) == 0 | n == Param.N
            if Param.MovieYes == 0
                s = (Param.N-n)/Param.N;
                Color ...
                = sqrt(s)*Param.ColorWhite + (1-sqrt(s))*Param.ColorBlack;
                pause
            end
        for m = 1:Param.M
            if Param.d == 1
                plot(Solution{n+1}(1,m),0, ...
                     Marker,'MarkerSize',Param.MarkerSize,'Color',Color)
                axis([Min{1},Max{1},-1,1])
                xlabel('First coordinate')
            end
            if Param.d == 2
                plot(Solution{n+1}(1,m),Solution{n+1}(2,m), ...
                     Marker,'MarkerSize',Param.MarkerSize,'Color',Color)
                axis([Min{1},Max{1},Min{2},Max{2}])
                xlabel('First coordinate')
                ylabel('Second coordinate')
            end
            if Param.d == 3
                plot3(Solution{n+1}(1,m),Solution{n+1}(2,m), ...
                      Solution{n+1}(3,m), ...
                      Marker,'MarkerSize',Param.MarkerSize,'Color',Color)
                axis([Min{1},Max{1},Min{2},Max{2},Min{3},Max{3}])
                xlabel('First coordinate')
                ylabel('Second coordinate')
                zlabel('Third coordinate')
            end
            hold on
        end
        grid on
        set(gca,'Color',Param.ColorLightBlue);
        set(gcf,'Color',Param.ColorLightBlue);
        set(gcf,'InvertHardCopy','off');
        set(gca,'FontSize',Param.FontSize);
        if Param.NovelApproachYes == 0
            title({[Title1,num2str(Result.Time(n+1))];Title2;[]});
        end
        if Param.NovelApproachYes == 1
            title({[Title1,num2str(Result.Time(n+1))];Title2;[]});
        end
        drawnow
        if Param.MovieYes == 1
            currFrame = getframe(gcf);
            writeVideo(vidObj,currFrame);
        end
        if Param.HoldOffYes == 1
            hold off
        end
    end
end
```

```
if Param.MovieYes == 0 & Param.SavePlotYes == 1
    saveas(gcf,File);
end

if Param.MovieYes == 1
    close(vidObj);
end

pause

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function AllPlots(FileP,FileV,Result,Param)

n0 = 1;
n1 = 1;
Param.MovieYes = 0;
Param.HoldOffYes = 0;
Param.SavePlotYes = 1;
Param = MinMaxPositionVelocity('P',Result,Param);
Param = MinMaxPositionVelocity('V',Result,Param);
PlotPositionVelocity('P',FileP,[0:n0:Param.N],Result,Param);
PlotPositionVelocity('V',FileV,[0:n1:Param.N],Result,Param);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function AllMovies(FileP,FileV,Result,Param)

Param.MovieYes = 1;
Param.HoldOffYes = 1;
Param = MinMaxPositionVelocity('P',Result,Param);
Param = MinMaxPositionVelocity('V',Result,Param);
MoviePositionVelocity('P',FileP,[0:Param.N],Result,Param);
MoviePositionVelocity('V',FileV,[0:Param.N],Result,Param);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Flock1234(TestCase,Flock,Param)

Param.NovelApproachYes = 0;
Param = SetParametersPlot(TestCase,Param);
Param = SetParametersCuckerSmale(Param);

if Flock == 1
    Param.d = 2;
    Param.M = 30;
    if Param.NovelApproachYes == 1
        Param.K0 = 20;
    end
```

```
    Param.IC = 'L';
    Param.RandomPerturbationYes = 1;
    Param.ScalingRandomPerturbationPosition = [1/5,1/5];
    Param.ScalingRandomPerturbationVelocity = [1/10,1/10];
    Param = SetParametersTimeIntegration(Param);
    Param = SetInitialCondition(Param);
    % Time integration
    tic
    [Result,Param] = TimeIntegration(Param);
    Time = toc
    % Plots
    FileP = 'MyPlot_CuckerSmale_Flock1_Position.jpg';
    FileV = 'MyPlot_CuckerSmale_Flock1_Velocity.jpg';
    AllPlots(FileP,FileV,Result,Param);
    % Movies
    % FileP = 'Movie_CuckerSmale_Flock1_Position.avi';
    % FileV = 'Movie_CuckerSmale_Flock1_Velocity.avi';
    % AllMovies(FileP,FileV,Result,Param);
end

if Flock == 2
    Param.d = 2;
    Param.M1 = 30;
    Param.M2 = 20;
    Param.M = Param.M1 + Param.M2;
    if Param.NovelApproachYes == 1
        Param.K0 = 20;
    end
    Param.IC = 'T';
    Param.RandomPerturbationYes = 1;
    Param.ScalingRandomPerturbationPosition = [1/5,1/5];
    Param.ScalingRandomPerturbationVelocity = [1/10,1/10];
    Param = SetParametersTimeIntegration(Param);
    Param = SetInitialCondition(Param);
    % Time integration
    [Result,Param] = TimeIntegration(Param);
    % Plots
    FileP = 'MyPlot_CuckerSmale_Flock2_Position.jpg';
    FileV = 'MyPlot_CuckerSmale_Flock2_Velocity.jpg';
    AllPlots(FileP,FileV,Result,Param);
    % Movies
    % FileP = 'Movie_CuckerSmale_Flock2_Position.avi';
    % FileV = 'Movie_CuckerSmale_Flock2_Velocity.avi';
    % AllMovies(FileP,FileV,Result,Param);
end

if Flock == 3
    Param.d = 3;
    Param.MFactor1 = 20;
    Param.MFactor2 = 5;
    Param.M = Param.MFactor1*Param.MFactor2;
    if Param.NovelApproachYes == 1
        Param.K0 = 10;
    end
    Param.IC = 'S';
```

```
    Param.RandomPerturbationYes = 1;
    Param.DirectionVelocity = [0,1,0];
    Param.ScalingRandomPerturbationPosition = [1/5,1/2,0];
    Param.ScalingRandomPerturbationVelocity = [1/10,1/10,0];
    Param = SetParametersTimeIntegration(Param);
    Param = SetInitialCondition(Param);
    % Time integration
    [Result,Param] = TimeIntegration(Param);
    % Plots
    FileP = 'MyPlot_CuckerSmale_Flock3_Position.jpg';
    FileV = 'MyPlot_CuckerSmale_Flock3_Velocity.jpg';
    AllPlots(FileP,FileV,Result,Param);
    % Movies
    % FileP = 'Movie_CuckerSmale_Flock3_Position.avi';
    % FileV = 'Movie_CuckerSmale_Flock3_Velocity.avi';
    % AllMovies(FileP,FileV,Result,Param);
end

if Flock == 4
    Param.d = 3;
    Param.MFactor1 = 20;
    Param.MFactor2 = 10;
    Param.M = Param.MFactor1*Param.MFactor2;
    if Param.NovelApproachYes == 1
        Param.K0 = 10;
    end
    Param.IC = 'S';
    Param.RandomPerturbationYes = 1;
    Param.DirectionVelocity = [1,1,1];
    Param.ScalingRandomPerturbationPosition = [1/5,1/5,0];
    Param.ScalingRandomPerturbationVelocity = [1/10,1/10,0];
    Param = SetParametersTimeIntegration(Param);
    Param = SetInitialCondition(Param);
    % Time integration
    [Result,Param] = TimeIntegration(Param);
    % Plots
    FileP = 'MyPlot_CuckerSmale_Flock4_Position.jpg';
    FileV = 'MyPlot_CuckerSmale_Flock4_Velocity.jpg';
    AllPlots(FileP,FileV,Result,Param);
    % Movies
    % FileP = 'Movie_CuckerSmale_Flock4_Position.avi';
    % FileV = 'Movie_CuckerSmale_Flock4_Velocity.avi';
    % AllMovies(FileP,FileV,Result,Param);
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```