SDE model

$$\mathrm{dR}(t) = \frac{-a}{4M^{1/4}} R(t).\,\mathrm{dt} + \frac{\sigma R(t)}{\alpha + R(t)} \mathrm{dW}(t)$$

where:

a : the growth parameter of each specific species

M : maximum body size

m_0 : initial mass of the organism at birth

σ, α : nonnegative constants controlling intensity of the white noise

$W(t)$: standard Wiener process

$R(t) = 1 - \left(\frac{m}{M}\right)^{\frac{1}{4}}$: the proprortion of total metabolic energy used for cell growth at time t

$r(t) = \left(\frac{m}{M}\right)^{\frac{1}{4}}$: the proprortion of total metabolic energy used for cell maintance at time t

Import data

```
species_dataset = xlsread("data_file.xls");
data = cell(1,13);
for i = 1:13
    data{i} = species_dataset(species_dataset(:,3)==i,:);
end
species = ["Guppy" "Heron" "Hen" "Salmon" "Rat" "Cod" "Robin" "Shrew" "Guinea Pig"
"Dom Rabbit" "Dom Pig" "Cow" "Shrimp"];
d = dictionary(species,data);
```

Fitted parameters

```
% % Get data of species Guppy
% v = values(d);
% % Parameters from West 2001 paper
% a = [0.1 1.56 0.47 0.026 0.23 0.017 1.9 0.83 0.21 0.36 0.31 0.28 0.027];
% m0 = [0.008 3 43 0.01 8 0.1 1 0.3 5 120 900 33300 0.0008];
% M = [0.15 2700 2100 2400 280 25000 22 4.2 840 1350 320000 442000 0.075];
% % sigma = [0.9301 0.0135 0.0989 1926 0.0647 0.0343 1.2311 0.0663 0.2847 0.0234
0.0191 0.0108 30.8249];
% % delta = [24.8794 0.0796 1.4949 29284 2.2108 1.1781 149.5158 0.4528 9.1656
1.7813 1.3943 3.6169 418.5605];
% sigma = [0.9301 0.2479 5.531 1000.9 4.7366 36.8374 1.3761 0.4572 0.5386 1.6795
5.9674 11.4047 30.8095];
% delta = [24.8794 5.1067 109.4324 21046 202.538 3018.9 163.5598 21.9439 61.5344
190.8999 201.3442 550.0898 418.3009];
% % Initial value R(0)
% R_initial = [];
```

```
% for i = 1:13
%     R_initial = [R_initial,1-(m0(i)/M(i))^(1/4)];
% end

% % Parameters (my version)
% a = [0.1196 1.6226 0.47 0.0159 0.2382 0.016 1.8367 0.7477 0.2142 0.3642 0.3324
0.2714 0.029];
% m0 = [0.0026 4.5158 38 0.0073 8 0.285 0.892 0.3 1.7816 105.9421 481.7243 33300
0.0008];
% M = [0.1482 2367.47 2100 2000 280.65 29000 22.878 4.5456 830.15 1354.63 311793.2
444489.3 0.0766];

% My parameters (round up)
a = [0.1 1.62 0.47 0.016 0.24 0.016 1.84 0.75 0.21 0.36 0.33 0.27 0.029];
m0 = [0.008 4.52 38 0.0073 8 0.29 0.89 0.3 1.78 106 482 33300 0.0008];
M = [0.15 2370 2100 2000 281 29000 22.88 4.55 830.15 1354.63 311800 444490 0.0766];
sigma = [0.93 0.26 5.54 121.24 3.32 33.83 0.087 4.28 1.11 1.12 5.97 15.53 30.22];
delta = [24.87 5.12 96.19 7033.5 113.3 3018.9 4.65 313.34 40.6 85.2 401.34 778.43
367];
sigma_0 = mean(sigma);
delta_0 = mean (delta);
% Initial value R(0)
R_initial = [];
for i = 1:13
    R_initial = [R_initial,1-(m0(i)/M(i))^(1/4)];
end
```

Execution options

```
deterministic_parameter_fit = 0;
fit_growth_rate = 0;
all_standard_deviation_fit = 0;
separate_standard_deviation_fit = 0;
dimensionless_deterministic_path = 0;
dimensionless_stochastic_path = 0;
universal_stochastic_path = 1;
separate_stochastic_path = 0;
```

Deterministic parameters fitting

```
if deterministic_parameter_fit
    % Get data of species Guppy
    v = values(d);
    data1 = v{13}
    % Initial values of parameters
    a = 0.01;
    m0 = 0.0005;
    M = 0.08;
    % Creat vectors of fitting parameters
    num_para = 3;
```

```
    para_fit_1 = zeros(1,num_para);
    para_fit_1(1) = a;
    para_fit_1(2) = m0;
    para_fit_1(3) = M;
    % Create the lower bound of fitting parameters
    lower_bound_1 = zeros(num_para,1);
    % Starting tim foe the fitting
    start_time = datetime("now")
    % Option for lsqcurvefit
    opts = optimoptions('lsqcurvefit','Algorithm','levenberg-marquardt');
    % Fit growth curve to data Guppy by using lsqcurvefit
    [fitted_para_1,resnorm_1,residual_1,exitflag_1,output_1] =
lsqcurvefit(@ontogenetic_growth,para_fit_1,data1(:,1),data1(:,2),...
        lower_bound_1,Inf(num_para,1),opts);
    % Print values of new fitted parameters
    fprintf('fitted parameters for data of species 1 - Guppy:\n')
    fprintf('a = %f, m0 = %f, M = %f \n', fitted_para_1(1:3))
    fprintf('RMSE: %f',sqrt(resnorm_1))
    % Compute the fitted growth curve
    tspan = linspace(0,data1(end,1)+6);
    fitted_growth_1 = ontogenetic_growth(fitted_para_1,tspan);
    % Show the time elasped for the fitting
    finish_time = datetime("now");
    Elasped_Time = finish_time - start_time;
end
```

Plot primitive data set for growth fit

```
if fit_growth_rate
    growth_time = [];
    growth_mass = [];
    % Plot
    for i =  1:13
        clf;
        growth_time = [growth_time,data{i}(:,1)];
        growth_mass = [growth_mass,data{i}(:,2)];
        plot(growth_time,growth_mass,'o','Linewidth',1.5)
        xlabel('Time (days)','FontSize',16)
        ylabel('Mass(g)','FontSize',16)
        title([species(i)],'FontSize',16);
        hold on
        x = linspace(0,max(growth_time),100);
        y = M(i)*(1-(1-(m0(i)/M(i)).^0.25)*exp(-a(i)*x/(4*M(i).^0.25))).^4;
        plot(x,y,'LineWidth',1.5)
        a_value = a(i);
        m0_value = m0(i);
        M_value = M(i);
        note_text = {['a = ',num2str(a_value)];['m_0 = ',num2str(m0_value)];['M =
',num2str(M_value)]};
```

```
        annotation('textbox',
[0.7,0.2,0.6,0.15],'String',note_text,'FitBoxToText','on','EdgeColor','black','Fonts
ize',12,'HorizontalAlignment','left')

        folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Deterministic
fit';
        filename = ['Deterministic_Growth',num2str(i),'.png'];
        saveas(gcf,fullfile(folder_path,filename));
        close(gcf);

        growth_time = [];
        growth_mass = [];
    end
end
```

Plot dimensionless growth pattern

```
if dimensionless_deterministic_path
    % Rescale data to tau and R(tau)
    for i = 1:13
    new_data{i} = [a(i)*data{i}(:,1)/(4*M(i).^(1/4))-log(1-(m0(i)/M(i)).^(1/4)),1-
(data{i}(:,2)/M(i)).^(1/4)];
    end
    newd = dictionary(species,new_data);
    newv = values(newd);
    x = linspace(0,9,500);
    y = exp(-x);
    % Plot the dimensionless-data
    marker = ['o' 's' 'd' 'p' '+' '*' 'x' '.' '^' 'v' '<' '>' 'h'];
    for i = 1:13
        plot(newv{i}(:,1),newv{i}(:,2),marker(i),'MarkerSize',5);
        hold on
    end
    hold on;
    plot(x,y,"LineWidth",1.5);
    xlabel('Dimensionless time, \tau','FontSize',16);
    ylabel('Dimensionless mass ratio, R','FontSize',16);
    ylim([0 1]);
    legend("Guppy", "Heron", "Hen", "Salmon", "Rat", "Cod", "Robin",
"Shrew", "Guinea Pig", "Dom Rabbit", "Dom Pig", "Cow", "Shrimp","Fitting
Curve",'Location','northeast','Orientation','vertical')
    % Save file
    folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Deterministic fit';
    filename = 'Dimensionless_Deterministic_Growth.png';
    saveas(gcf,fullfile(folder_path,filename));
    close(gcf);
end
```

Plot standard deviation for all species

```
if all_standard_deviation_fit
    % Rescale all data to tau and R(tau)
    new_data = cell(1,13);
    for i = 1:13
        new_data{i} = [(a(i)*data{i}(:,1))/(4*M(i)^0.25)-log(R_initial(i)),1-
(data{i}(:,2)/M(i)).^(1/4)];
    end
    % Find standard deviatioin
    standard_deviation = cell(1,13);
    base_para = cell(1,13);
    for i = 1:13
        base_para{i} = [R_initial(i),a(i),M(i)];
        standard_deviation{i} = [new_data{i}(:,1),abs(new_data{i}(:,2)-exp(-
new_data{i}(:,1)))];
    end
    % store data for fitting app
    time = [];
    mass = [];
    for i = 1:13
       time = [time;standard_deviation{i}(:,1)];
       mass = [mass;standard_deviation{i}(:,2)];
    end
    % The function of noise intensify
    y = @(x)(0.045) ./ (0.9667 + x);
    x_value = linspace(0,9,100);
    y_value = y(x_value);
    % Plot
    marker = ['o' 's' 'd' 'p' '+' '*' 'x' '.' '^' 'v' '<' '>' 'h'];
    for i = 1:13
        plot(standard_deviation{i}(:,1),standard_deviation{i}
(:,2),marker(i),"MarkerSize",5)
        hold on
    end
    hold on
    plot(x_value,y_value,"LineWidth",1.5,'Color','red');
    legend("Guppy", "Heron", "Hen","Salmon", "Rat", "Cod", "Robin",
"Shrew", "Guinea Pig", "Dom Rabbit", "Dom Pig", "Cow","Shrimp","Fitting
Curve",'Location','northeast','Orientation','vertical')
    xlabel('Dimensionless time, \tau','FontSize',16);
    ylabel('The standard deviation','FontSize',16);
    % Save file
    folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Deterministic fit';
    filename = 'Dimensionless_Standard_deviation.png';
    saveas(gcf,fullfile(folder_path,filename));
    close(gcf);
end
```

Plot standard deviation for each group

```
if separate_standard_deviation_fit
```

```
    % Rescale all data to R(t), not time
    new_data = cell(1,13);
    for i = 1:13
        new_data{i} = [data{i}(:,1),1-(data{i}(:,2)/M(i)).^(1/4)];
    end
    % Find standard deviation
    standard_deviation = cell(1,13);
    base_para = cell(1,13);
    for i = 1:13
        base_para{i} = [R_initial(i),a(i),M(i)];
        standard_deviation{i} = [new_data{i}(:,1),abs(new_data{i}(:,2)-
R_initial(i)*exp((-a(i)*new_data{i}(:,1))/(4*M(i)^0.25)))];
    end
    % Store data for fitting app
    time = [];
    mass = [];
    for i = 1:13
        time = [time;standard_deviation{i}(:,1)];
        mass = [mass;standard_deviation{i}(:,2)];
    end

    time_sp = [standard_deviation{6}(:,1)];
    mass_sp = [standard_deviation{6}(:,2)];

    growth_time = [];
    growth_mass = [];
    % Plot
    for i = 6
        clf;
        growth_time = [growth_time,standard_deviation{i}(:,1)];
        growth_mass = [growth_mass,standard_deviation{i}(:,2)];
        plot(growth_time,growth_mass,'o','Linewidth',1.5)
        xlabel('Time (days)','FontSize',16)
        ylabel('The standard deviation','FontSize',16)
        ylim([0 0.15]);
        title([species(i)],'FontSize',16);
        hold on
        x = linspace(0,max(growth_time)+max(growth_time)/10,100);
        y = sigma(i)./(delta(i)+x);
        plot(x,y,'LineWidth',1.5)
        sigma_value = sigma(i);
        delta_value = delta(i);
        note_text = {['\sigma = ',num2str(sigma_value)];['\delta =
',num2str(delta_value)]};
        annotation('textbox',
[0.65,0.2,0.1,0.7],'String',note_text,'FitBoxToText','on','EdgeColor','black','Fonts
ize',12,'HorizontalAlignment','left')

        folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Diffussion fit';
        filename = ['Diffussion_fit',num2str(i),'.png'];
```

```
        saveas(gcf,fullfile(folder_path,filename));
        close(gcf);

        growth_time = [];
        growth_mass = [];
    end
end
```

Plot solution paths

```
if separate_stochastic_path
    % Rescale R not time
    new_data = cell(1,13);
    for i = 1:13
        new_data{i} = [data{i}(:,1),1-(data{i}(:,2)/M(i)).^(1/4)];
    end

    % Plot
    for i = 6
            clf;
            plot(new_data{i}(:,1),new_data{i}(:,2),'.','MarkerSize',20,'Color','r')
            xlabel('Time (days), t','FontSize',16)
            ylabel('Metabolism for cell growth, R','FontSize',16)
            title([species(i)],'FontSize',16);
            a_value = a(i);
            m0_value = m0(i);
            M_value = M(i);
            sigma_value = sigma(i);
            delta_value = delta(i);
            note_text = {['a = ',num2str(a_value)];['m_0 = ',num2str(m0_value)];
['M = ',num2str(M_value)];['\sigma = ',num2str(sigma_value)];['\delta =
',num2str(delta_value)]};
            annotation('textbox',
[0.65,0.2,0.1,0.7],'String',note_text,'FitBoxToText','on','EdgeColor','black','Fonts
ize',12,'HorizontalAlignment','left')
            hold on
        for n = 1:10
            randompick = randi([1,1000],1);
            T1 = 0;
            T2 = round(max(new_data{i}(:,1)))+5;
            Dt = 5*10^(-3);
            rng(randompick);
            [Time,xrk] =
rk_stochastic(a(i),M(i),sigma(i),delta(i),T1,T2,Dt,R_initial(i));
            xrk1 = [R_initial(i),xrk];
            plot(Time,xrk1,'b--','Linewidth',1.5)
            hold on
        end
            folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Stochastic
Path';
```

```
            filename = ['Stochastic Path',num2str(i),'.png'];
            saveas(gcf,fullfile(folder_path,filename));
            close(gcf);
            % % Fix diffussion term
            % folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Fix
diffussion term';
            % filename = ['Stochastic Path Fix',num2str(i),'.png'];
            % saveas(gcf,fullfile(folder_path,filename));
            % close(gcf);
    end
end
```

Plot universal stochastic path (Runge _kutta)

```
if dimensionless_stochastic_path
    % Rescale all data to tau and R(tau)
    new_data = cell(1,13);
    for i = 1:13
        new_data{i} = [(a(i)*data{i}(:,1))/(4*M(i)^0.25)-log(R_initial(i)),1-
(data{i}(:,2)/M(i)).^(1/4)];
    end
    % Plot dimensionless curves
    for n = 1:5
        randompick = randi([1,1000],1);
        T1 = 0;
        T2 = 10;
        Dt = 5*10^(-3);
        rng(randompick);
        [Time,xrk] = rk_stochastic(4,1,0.3,1,T1,T2,Dt,0.9);
        xrk1 = [R_initial(i),xrk];
        plot(Time,xrk1,'g--','Linewidth',1.5)
        hold on
        legend('off')
    end
    marker = ['o' 's' 'd' 'p' '+' '*' 'x' '.' '^' 'v' '<' '>' 'h'];
    for i = 1:13
        plot(new_data{i}(:,1),new_data{i}
(:,2),marker(i),'MarkerSize',5,'DisplayName',species(i));
        hold on
    end
    xlabel('Dimensionless time, \tau','FontSize',16)
    ylabel('Dimentionless mass ratio, R','FontSize',16)
    ylim([0,1]);
    legend('show','Location','northeast','Orientation','vertical')

    folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Stochastic Path';
    filename = 'Dimensionless Stochastic Path.png';
    saveas(gcf,fullfile(folder_path,filename));
    close(gcf);
end
```

Plot Universal Stochastic Path (Brownian Motion codes)

```
if universal_stochastic_path
    % Rescale all data to tau and R(tau)
    new_data = cell(1,13);
    for i = 1:13
        new_data{i} = [(a(i)*data{i}(:,1))/(4*M(i)^0.25)-log(R_initial(i)),1-
(data{i}(:,2)/M(i)).^(1/4)];
    end
    newd = dictionary(species,new_data);
    newv = values(newd);
    figure;
    % Plot dimensionless stochastic curves
    nPeriods = 11
    dt = 1;
    BM = bm(0,0.3,'StartState',0);
    [Paths,timeseries] = simulate(BM,nPeriods,'DeltaTime',dt,'nTrials',50000);
    for i = 1:20
        plot(timeseries,exp(-timeseries+Paths(:,:,i)), 'HandleVisibility', 'off')
        hold on
    end

   marker = ['o' 's' 'd' 'p' '+' '*' 'x' '.' '^' 'v' '<' '>' 'h'];
    for i = 1:13
        plot(newv{i}(:,1),newv{i}
(:,2),marker(i),'MarkerSize',5,'DisplayName',species(i));
        hold on
    end
    legend('show','Location','northeast','Orientation','vertical')
    xlabel('Dimensionless time, T','FontSize',16);
    ylabel('Dimensionless mass ratio, R','FontSize',16);
    ylim([0 1]);

    folder_path = 'C:\Users\hgmchau\Desktop\13 species data fit\Stochastic Path';
    filename = 'Dimensionless Stochastic Path.png';
    saveas(gcf,fullfile(folder_path,filename));
end
```

```
nPeriods = 11
```

Local function - Growth curve

```
function m = ontogenetic_growth(para_fit,t)
a = para_fit(1);
m0 = para_fit(2);
M = para_fit(3);
m = M*(1-(1-(m0/M)^(1/4))*exp(-a*t/(4*M^(1/4)))).^4;
end
```

Local function - Runge-Kutta algorithm

```
function [Time,xrk] = rk_stochastic(a,M,sigma,alpha,T1,T2,Dt,R_initial)
    T = T2-T1;
    n = T/Dt;
    dt = (Dt)^2;
    N = Dt/dt;

    Time = zeros(1,n+1);
    xrk = zeros(1,n);
    xtilde = zeros(1,3);
    xfinal = R_initial;

    for k = 1:n
        x1 = 0;
        for m = 1:N
            dW1 = sqrt(dt)*randn;
            x1 = x1 + dW1;
        end
        x = xfinal;
        xtilde(1) = x + f_drift(x,a,M)*Dt;
        xtilde(2) = xtilde(1)+ f_diffusion(x,sigma,alpha)*(x1^2-Dt)/(2*sqrt(Dt));
        xtilde(3) = xtilde(1)- f_diffusion(x,sigma,alpha)*(x1^2-Dt)/(2*sqrt(Dt));
        xfinal = x + 0.5*Dt*(f_drift(x,a,M)+f_drift(xtilde(1),a,M))...
                  + x1*f_diffusion(x,sigma,alpha)...
                  + 0.5*sqrt(Dt)*(f_diffusion(xtilde(2),sigma,alpha)-
f_diffusion(xtilde(3),sigma,alpha));
        xrk(k) = xfinal;
        Time(k+1) = k*Dt;
    end
end
```

Local function - Drift term

```
function y = f_drift(x,a,M)
% For each species
y = -a/(4*M^0.25)*x;
% % For dimensionless curve, choose a = 4 and M = 1
end
```

Local function - Difussion term

```
function y = f_diffusion(x,sigma,delta)
% For each species
y = sigma*x/(delta+x);
end
```

Local function - Deterministic Fit

```
% function m = ontogenetic_growth(para_fit,t)
```

```
% a = para_fit(1);
% m0 = para_fit(2);
% M = para_fit(3);
% m = M*(1-(1-(m0/M)^(1/4))*exp(-a*t/(4*M^(1/4)))).^4;
% end
```