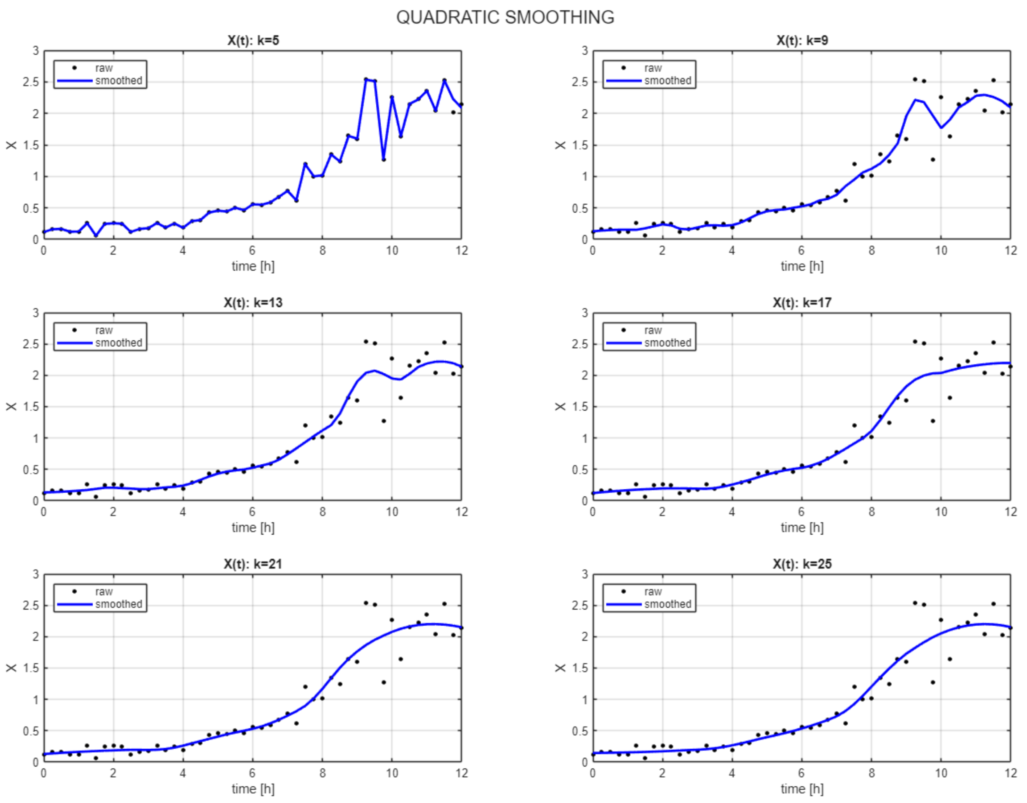
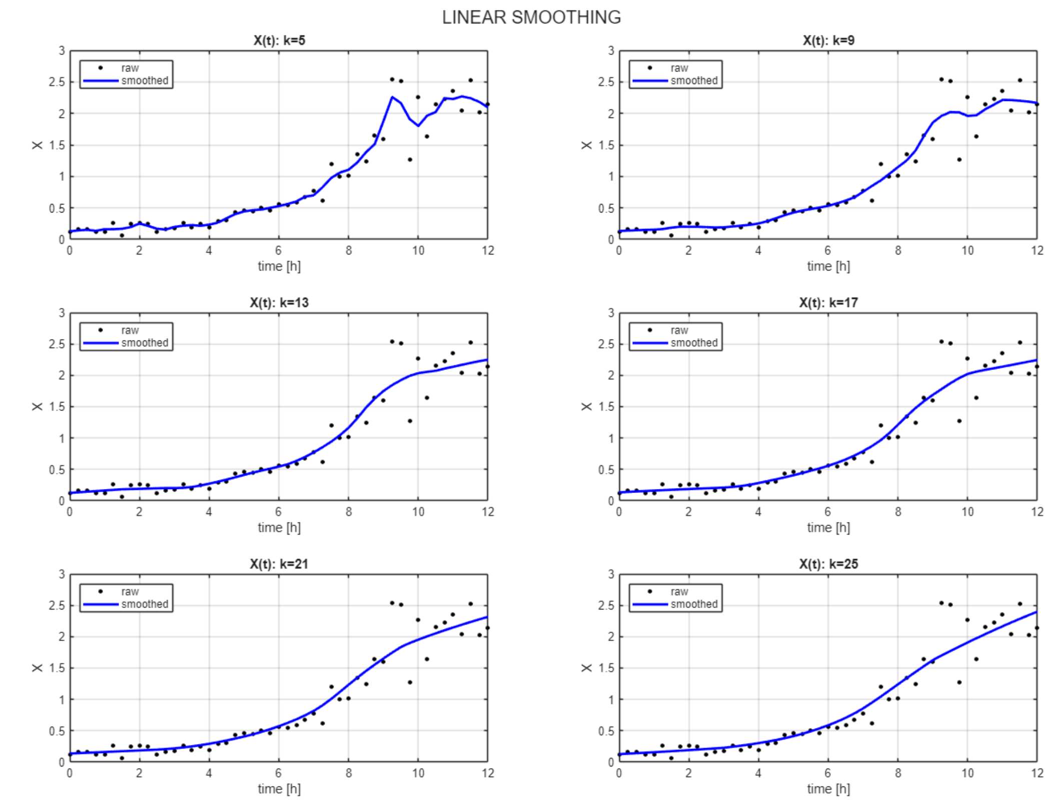
Table of Contents

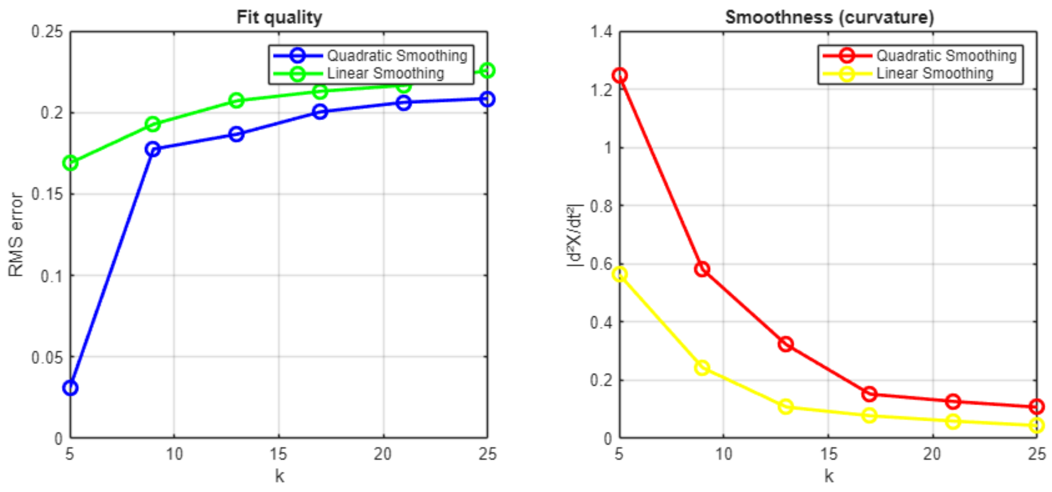
Section 1: Smoothing Analysis

```
clear; close all;

% Load data_set_2 (save as data_set_2.mat first, or hardcode)
load('data_set_2.mat');  % gives t, bio_r
X = bio_r;  % rename for clarity
t = time;

% Test multiple k values
ks = [5 9 13 17 21 25];
figure('Position', [100 100 1400 1000]);

rms_errors = zeros(length(ks),2);
curvature = zeros(length(ks),2);

sgtitle("QUADRATIC SMOOTHING")
for i = 1:length(ks)
    subplot(3,2,i);
    X_smooth = local_smoothing(t, X, ks(i));  % Single call per subplot
    plot(t, X, 'ko', 'MarkerSize', 3, 'MarkerFaceColor', 'k'); hold on;
    plot(t, X_smooth, 'b-', 'LineWidth', 2);
    title(sprintf('X(t): k=%d', ks(i)));
    xlabel('time [h]'); ylabel('X');
    legend('raw', 'smoothed', 'Location','northwest');
    grid on;

    rms_errors(i,1) = sqrt(mean((X_smooth - X).^2));
    dX = numerical_diff(t, X_smooth, '2');
    d2X = numerical_diff(t, dX, '2');
    curvature(i,1) = mean(abs(d2X));
end

figure('Position', [100 100 1400 1000]);
sgtitle("LINEAR SMOOTHING")
for i = 1:length(ks)
    subplot(3,2,i);
    X_smooth2 = local_smoothing_linear(t, X, ks(i));  % Single call per
subplot
    plot(t, X, 'ko', 'MarkerSize', 3, 'MarkerFaceColor', 'k'); hold on;
    plot(t, X_smooth2, 'b-', 'LineWidth', 2);
    title(sprintf('X(t): k=%d', ks(i)));
    xlabel('time [h]'); ylabel('X');
    legend('raw', 'smoothed', 'Location','northwest');
    grid on;
    rms_errors(i,2) = sqrt(mean((X_smooth2 - X).^2));
    dX2 = numerical_diff(t, X_smooth2, '2');
    d2X2 = numerical_diff(t, dX2, '2');
    curvature(i,2) = mean(abs(d2X2));
end

figure('Position', [200 200 1000 400]);
subplot(1,2,1);
plot(ks, rms_errors(:,1), 'bo-', 'LineWidth', 2, 'MarkerSize', 8);
hold on
plot(ks, rms_errors(:,2), 'go-', 'LineWidth', 2, 'MarkerSize', 8);
legend("Quadratic Smoothing","Linear Smoothing")
xlabel('k'); ylabel('RMS error'); title('Fit quality');
grid on;

subplot(1,2,2);
plot(ks, curvature(:,1), 'ro-', 'LineWidth', 2, 'MarkerSize', 8);
hold on
plot(ks, curvature(:,2), 'yo-', 'LineWidth', 2, 'MarkerSize', 8);
xlabel('k'); ylabel('|d²X/dt²|'); title('Smoothness (curvature)');
legend("Quadratic Smoothing","Linear Smoothing")
grid on;



fprintf('k\tRMS_Q\tRMS_L\tCurvature_Q\tCurvature_L\n');
fprintf('-------------------\n');
for i=1:length(ks)
    fprintf('%d\t%.4f\t%.4f\t%.4f\t%.4f\n', ks(i),
rms_errors(i,1),rms_errors(i,2), curvature(i,1),curvature(i,2));
end

curvature_raw = mean(abs(numerical_diff(t, numerical_diff(t,X,2),2)));
fprintf('Raw curvature = %.4f\n', curvature_raw);
```

```
k    RMS_Q    RMS_L    Curvature_Q    Curvature_L
-------------------
5    0.0308    0.1689    1.2476    0.5638
9    0.1775    0.1928    0.5800    0.2428
13    0.1867    0.2072    0.3221    0.1080
17    0.2004    0.2130    0.1524    0.0782
21    0.2062    0.2167    0.1272    0.0597
25    0.2086    0.2260    0.1076    0.0443
Raw curvature = 1.4761
```


QUADRATIC SMOOTHING
X(t): k=5
X(t): k=9
X(t): k=13
X(t): k=17
X(t): k=21
X(t): k=25
raw
smoothed
X
time [h]





Section 2: Numerical Differentiation Order comparison

```
% Optimal parameters from your analysis
k_opt = 17;

order2 = numerical_diff(t, X, '2'); mu2 = order2 ./ X;
```

```
order3 = numerical_diff(t, X, '3'); mu3 = order3 ./ X;
log_mask = t>=4 & t<=10;

fprintf('Order 2: std(μ)=%.3f\n', std(mu2(log_mask)));
fprintf('Order 3: std(μ)=%.3f\n', std(mu3(log_mask)));

X_smooth = local_smoothing(t, X, k_opt);

order2 = numerical_diff(t, X_smooth, '2'); mu2 = order2 ./ X_smooth;
order3 = numerical_diff(t, X_smooth, '3'); mu3 = order3 ./ X_smooth;
log_mask = t>=4 & t<=10;

fprintf('Order 2: std(μ)=%.3f\n', std(mu2(log_mask)));
fprintf('Order 3: std(μ)=%.3f\n', std(mu3(log_mask)));

X_smooth_linear = local_smoothing_linear(t, X, k_opt);

order2 = numerical_diff(t, X_smooth_linear, '2'); mu2 = order2 ./
X_smooth_linear;
order3 = numerical_diff(t, X_smooth_linear, '3'); mu3 = order3 ./
X_smooth_linear;
log_mask = t>=4 & t<=10;

fprintf('Order 2: std(μ)=%.3f\n', std(mu2(log_mask)));
fprintf('Order 3: std(μ)=%.3f\n', std(mu3(log_mask)));
```

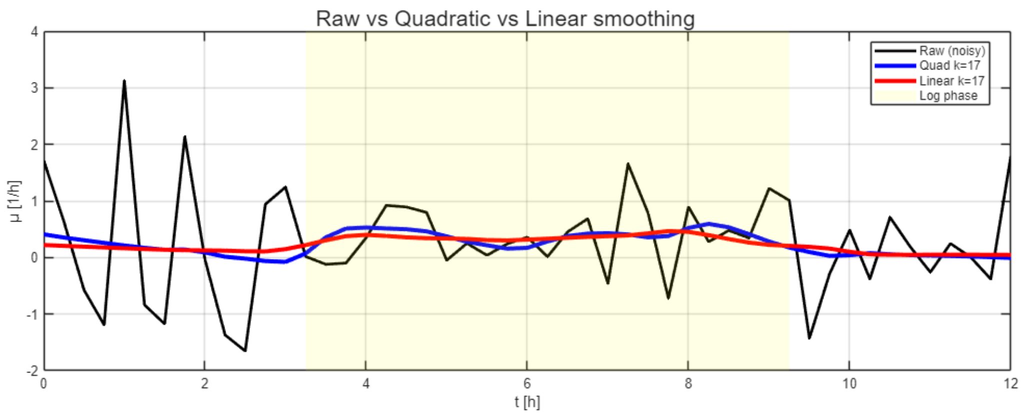
```
Order 2: std(μ)=0.502
Order 3: std(μ)=0.649
Order 2: std(μ)=0.157
Order 3: std(μ)=0.162
Order 2: std(μ)=0.090
Order 3: std(μ)=0.091
```

Section 3: Actual Growth Rate calculation

```
X_raw = X;

% 1. RAW μ(t)
dX_raw = numerical_diff(t, X_raw, '3');
mu_raw = dX_raw ./ X_raw;

% 2. QUADRATIC SMOOTHING μ(t)
X_smooth_quad = local_smoothing(t, X_raw, k_opt);  % Your quadratic function
dX_quad = numerical_diff(t, X_smooth_quad, '3');
mu_quad = dX_quad ./ X_smooth_quad;

% 3. LINEAR SMOOTHING μ(t)
X_smooth_lin = local_smoothing_linear(t, X_raw, k_opt);  % You'll need this
function
dX_lin = numerical_diff(t, X_smooth_lin, '3');
mu_lin = dX_lin ./ X_smooth_lin;

% Plot all three
```

```
figure('Position', [100 100 1200 400]);
plot(t, mu_raw, 'k-', 'LineWidth', 2); hold on;
plot(t, mu_quad, 'b-', 'LineWidth', 3);
plot(t, mu_lin, 'r-', 'LineWidth', 3);

% FIXED mask → indices
log_mask = t >= 3.25 & t <= 9.25;
log_start = find(log_mask, 1, 'first');   % FIRST true index
log_end   = find(log_mask, 1, 'last');    % LAST true index

% Yellow highlight
fill([t(log_start) t(log_end) t(log_end) t(log_start)], [-2,-2,4,4], 'y',
'FaceAlpha', 0.1, 'EdgeColor', 'none');

xlabel('t [h]'); ylabel('μ [1/h]');
legend('Raw (noisy)', sprintf('Quad k=%d', k_opt), sprintf('Linear k=%d',
k_opt), ...
       'Log phase','Location','best');
grid on;
% title('Task 3.2: μ(t) comparison');

sgtitle('Raw vs Quadratic vs Linear smoothing', 'FontSize', 16);

% Report AVERAGES (log-phase t=4-10 typical)

fprintf('Task 3.2 μ(t) AVERAGES (t=4-10):\n');
fprintf('Raw:     %.4f ± %.4f\n', mean(mu_raw(log_mask)),
std(mu_raw(log_mask)));
fprintf('Quad k=%d: %.4f ± %.4f\n', k_opt, mean(mu_quad(log_mask)),
std(mu_quad(log_mask)));
fprintf('Linear k=%d: %.4f ± %.4f\n', k_opt, mean(mu_lin(log_mask)),
std(mu_lin(log_mask)));
```

```
Task 3.2 μ(t) AVERAGES (t=4-10):
Raw:     0.4098 ± 0.5387
Quad k=17: 0.3732 ± 0.1386
Linear k=17: 0.3424 ± 0.0671
```



Published with MATLAB® R2025b