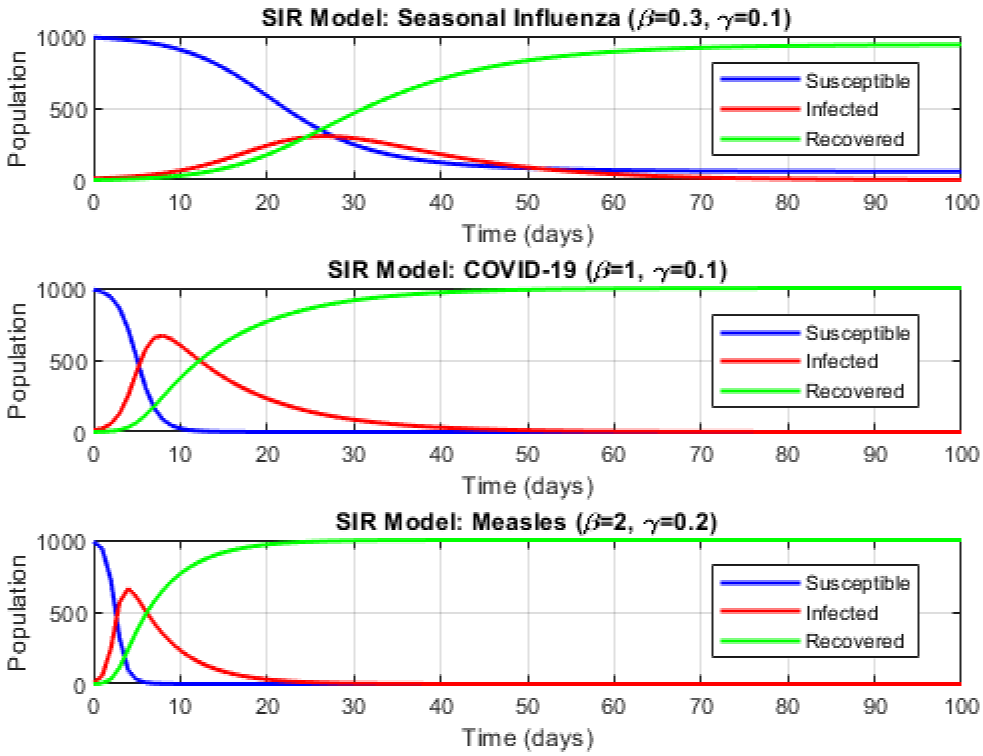
```matlab
%Initial Parameters
h = 1;                  %Time step (in days)
T = 100;                %Total simulation time (in days)
N = 1000;               %Total population
t = 0:h:T;              %Time vector

%Initial Conditions
S0 = 990;               %Initial susceptible people
I0 = 10;                %Initial infected people
R0 = 0;                 %Initial recovered people

%Parameter Combinations: [beta, gamma]
param_sets = [
    0.3, 0.1;           %Seasonal Influenza
    1.0, 0.1;           %COVID-19
    2.0, 0.2            %Measles
];
param_names = {'Seasonal Influenza', 'COVID-19', 'Measles'};

%Solve SIR model for each parameter set
figure;
for scenario = 1:size(param_sets, 1)
    beta = param_sets(scenario, 1);
    gamma = param_sets(scenario, 2);

    %Initialize State Vars
    S = zeros(1, length(t));
    I = zeros(1, length(t));
    R = zeros(1, length(t));
    S(1) = S0;
    I(1) = I0;
    R(1) = R0;

    %Runge-Kutta 4th Order
    for i = 1:(length(t)-1)
        %Current state
        S_i = S(i);
        I_i = I(i);
        R_i = R(i);

        %Define Derivatives
        f1_S = @(S, I) -beta / N * S * I;
        f1_I = @(S, I) beta / N * S * I - gamma * I;
        f1_R = @(I) gamma * I;

        %RK4 steps for S
        k1_S = h * f1_S(S_i, I_i);
        k1_I = h * f1_I(S_i, I_i);
        k1_R = h * f1_R(I_i);

        k2_S = h * f1_S(S_i + k1_S / 2, I_i + k1_I / 2);
        k2_I = h * f1_I(S_i + k1_S / 2, I_i + k1_I / 2);
```

```
        k2_R = h * f1_R(I_i + k1_I / 2);

        k3_S = h * f1_S(S_i + k2_S / 2, I_i + k2_I / 2);
        k3_I = h * f1_I(S_i + k2_S / 2, I_i + k2_I / 2);
        k3_R = h * f1_R(I_i + k2_I / 2);

        k4_S = h * f1_S(S_i + k3_S, I_i + k3_I);
        k4_I = h * f1_I(S_i + k3_S, I_i + k3_I);
        k4_R = h * f1_R(I_i + k3_I);

        %Update Values
        S(i+1) = S_i + (k1_S + 2*k2_S + 2*k3_S + k4_S) / 6;
        I(i+1) = I_i + (k1_I + 2*k2_I + 2*k3_I + k4_I) / 6;
        R(i+1) = R_i + (k1_R + 2*k2_R + 2*k3_R + k4_R) / 6;
    end

    %Plotting
    subplot(3, 1, scenario);
    plot(t, S, 'b', 'LineWidth', 1.5, 'DisplayName', 'Susceptible');
    hold on;
    plot(t, I, 'r', 'LineWidth', 1.5, 'DisplayName', 'Infected');
    plot(t, R, 'g', 'LineWidth', 1.5, 'DisplayName', 'Recovered');
    title(['SIR Model: ', param_names{scenario}, ' (\beta=', num2str(beta),
', \gamma=', num2str(gamma), ')']);
    xlabel('Time (days)');
    ylabel('Population');
    legend('Location', 'best');
    grid on;
end
```



Published with MATLAB® R2024a

Discussion 1:

The impacts of β and γ are that higher transmission rates (β) lead to faster disease spread while higher recovery rates (γ) reduce the max levels of infection and lead to a quicker end to the outbreak.

Seasonal Influenza has a moderate β of 0.3 and moderate γ of 0.1, so the infection spreads gradually, with a moderate peak.

COVID-19 has a higher β of 1 with still moderate γ of 0.1, so there's a faster spread and larger peak in infections and slower recovery rate, relatively speaking.

Measles has a very high β of 2 and somewhat higher γ of 0.2, so there's an even faster spread but now with a somewhat faster resolution.

Note: Discussion Part 2 and Discussion Part 3 are within the output of the code.

Part 2 Interpolation

```
h_fine = 1;
h_coarse = 2;
T = 100;
N = 1000;
t_fine = 0:h_fine:T;
t_coarse = 0:h_coarse:T;
t_odd = 1:2:T;

% Initial Conditions
S0 = 990;
I0 = 10;
R0 = 0;

% seasonal influenza
beta = 0.3;
gamma = 0.1;

% Function
function [S, I, R] = simulate_SIR(t, h, S0, I0, R0, beta, gamma, N)
    S = zeros(1, length(t));
    I = zeros(1, length(t));
    R = zeros(1, length(t));
    S(1) = S0; I(1) = I0; R(1) = R0;

    for i = 1:(length(t) - 1)
        S_i = S(i); I_i = I(i); R_i = R(i);

        % Derivatives
        f1_S = @(S, I) -beta / N * S * I;
        f1_I = @(S, I) beta / N * S * I - gamma * I;
        f1_R = @(I) gamma * I;

        % RK4 steps
        k1_S = h * f1_S(S_i, I_i);
        k1_I = h * f1_I(S_i, I_i);
        k1_R = h * f1_R(I_i);

        k2_S = h * f1_S(S_i + k1_S / 2, I_i + k1_I / 2);
        k2_I = h * f1_I(S_i + k1_S / 2, I_i + k1_I / 2);
        k2_R = h * f1_R(I_i + k1_I / 2);

        k3_S = h * f1_S(S_i + k2_S / 2, I_i + k2_I / 2);
        k3_I = h * f1_I(S_i + k2_S / 2, I_i + k2_I / 2);
        k3_R = h * f1_R(I_i + k2_I / 2);

        k4_S = h * f1_S(S_i + k3_S, I_i + k3_I);
        k4_I = h * f1_I(S_i + k3_S, I_i + k3_I);
        k4_R = h * f1_R(I_i + k3_I);

        % Updated values
```

```
        S(i + 1) = S_i + (k1_S + 2 * k2_S + 2 * k3_S + k4_S) / 6;
        I(i + 1) = I_i + (k1_I + 2 * k2_I + 2 * k3_I + k4_I) / 6;
        R(i + 1) = R_i + (k1_R + 2 * k2_R + 2 * k3_R + k4_R) / 6;
    end
end


[S_fine, I_fine, R_fine] = simulate_SIR(t_fine, h_fine, S0, I0, R0, beta,
gamma, N);
[S_coarse, I_coarse, R_coarse] = simulate_SIR(t_coarse, h_coarse, S0, I0,
R0, beta, gamma, N);

% Linear Interpolation
S_linear_interp = interp1(t_coarse, S_coarse, t_odd, 'linear');
I_linear_interp = interp1(t_coarse, I_coarse, t_odd, 'linear');
R_linear_interp = interp1(t_coarse, R_coarse, t_odd, 'linear');

% Quadratic Interpolation
function indices = nearest_indices(array, target, num_points)
    [~, sorted_indices] = sort(abs(array - target));
    indices = sorted_indices(1:num_points);
    indices = sort(indices);
end

function values = nearest_points(array, target, num_points)
    indices = nearest_indices(array, target, num_points);
    values = array(indices);
end

function V_interp = quadratic_interp(t_points, V_points, t_interp)
    n = length(t_points);
    V_interp = 0;
    for i = 1:n
        L = 1;
        for j = 1:n
            if i ~= j
                L = L .* (t_interp - t_points(j)) / (t_points(i) -
t_points(j));
            end
        end
        V_interp = V_interp + V_points(i) * L;
    end
end

S_quadratic_interp = arrayfun(@(t) ...
    quadratic_interp(nearest_points(t_coarse, t, 3), ...
                     S_coarse(nearest_indices(t_coarse, t, 3)), t), ...
    t_odd);

I_quadratic_interp = arrayfun(@(t) ...
    quadratic_interp(nearest_points(t_coarse, t, 3), ...
                     I_coarse(nearest_indices(t_coarse, t, 3)), t), ...
    t_odd);
```

```
R_quadratic_interp = arrayfun(@(t) ...
    quadratic_interp(nearest_points(t_coarse, t, 3), ...
                     R_coarse(nearest_indices(t_coarse, t, 3)), t), ...
    t_odd);

% True Values
odd_indices = t_odd / h_fine + 1;
S_true = S_fine(odd_indices);
I_true = I_fine(odd_indices);
R_true = R_fine(odd_indices);

% L2 Errors
EL2_S_linear = sqrt(sum((S_linear_interp - S_true).^2) / length(t_odd));
EL2_I_linear = sqrt(sum((I_linear_interp - I_true).^2) / length(t_odd));
EL2_R_linear = sqrt(sum((R_linear_interp - R_true).^2) / length(t_odd));

EL2_S_quad = sqrt(sum((S_quadratic_interp - S_true).^2) / length(t_odd));
EL2_I_quad = sqrt(sum((I_quadratic_interp - I_true).^2) / length(t_odd));
EL2_R_quad = sqrt(sum((R_quadratic_interp - R_true).^2) / length(t_odd));

disp('L2 Errors for Interpolation Methods:');
errors = table([EL2_S_linear; EL2_S_quad], ...
               [EL2_I_linear; EL2_I_quad], ...
               [EL2_R_linear; EL2_R_quad], ...
               'VariableNames', {'Susceptible', 'Infected', 'Recovered'}, ...
               'RowNames', {'Linear', 'Quadratic'});
disp(errors);

% results
disp('Interpolation Comparison:');
if sum(errors{1, :}) < sum(errors{2, :})
    disp('Linear interpolation has smaller errors overall.');
else
    disp('Quadratic interpolation has smaller errors overall.');
end
```

```
L2 Errors for Interpolation Methods:
                 Susceptible    Infected    Recovered
                 ___________    ________    _________

    Linear         0.65133       0.50414     0.41293
    Quadratic       0.1161      0.098859    0.061403

Interpolation Comparison:
Quadratic interpolation has smaller errors overall.
```

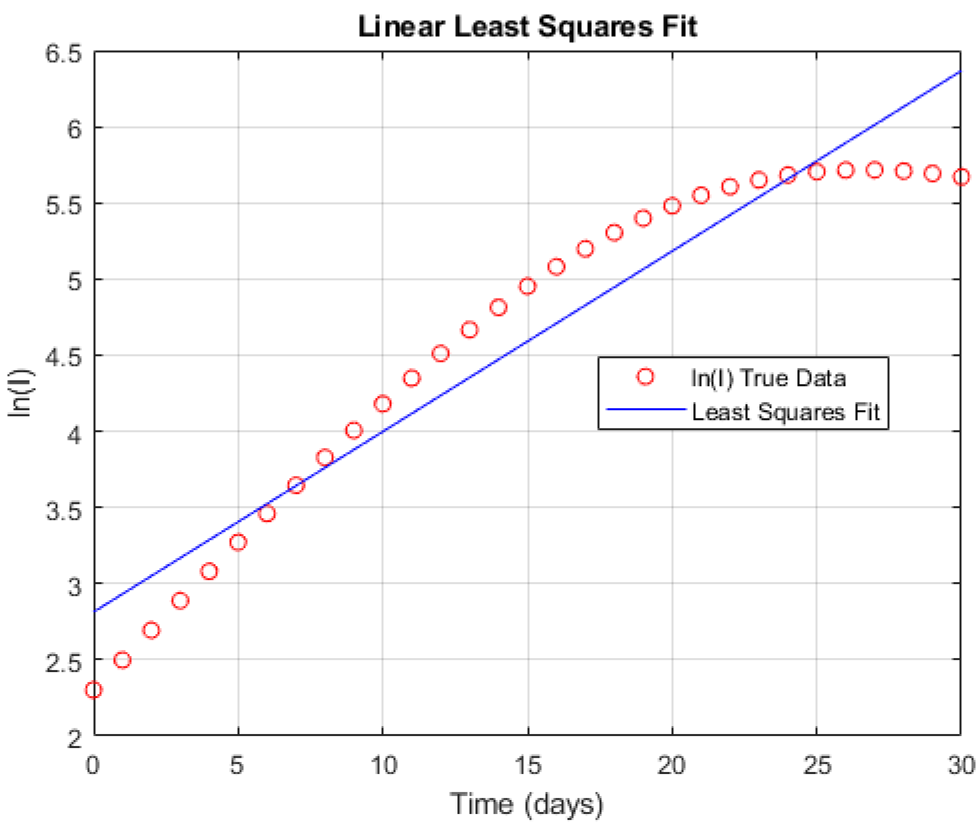
Published with MATLAB® R2024a

```
% MAE384 Final Project Part III
% Rudy Medrano 1221389923
clc; clear; close all;

% Initial Parameters (same as Part I)
h = 1;                 % Time step (in days)
T = 30;                % Total simulation time (30 days for Part III)
N = 1000;              % Total population
t = 0:h:T;             % Time vector

% Initial Conditions
S0 = 990;              % Initial susceptible people
I0 = 10;               % Initial infected people
R0 = 0;                % Initial recovered people

% Parameters for Part III
beta = 0.3;            % Transmission rate
gamma = 0.1;           % Recovery rate

% Generate "True" Data with RK4
S = zeros(1, length(t));
I = zeros(1, length(t));
R = zeros(1, length(t));
S(1) = S0;
I(1) = I0;
R(1) = R0;

% Runge-Kutta 4th Order
for i = 1:(length(t)-1)
    % Current state
    S_i = S(i);
    I_i = I(i);
    R_i = R(i);

    % Define Derivatives
    f1_S = @(S, I) -beta / N * S * I;
    f1_I = @(S, I) beta / N * S * I - gamma * I;
    f1_R = @(I) gamma * I;

    % RK4 steps for S
    k1_S = h * f1_S(S_i, I_i);
    k1_I = h * f1_I(S_i, I_i);
    k1_R = h * f1_R(I_i);

    k2_S = h * f1_S(S_i + k1_S / 2, I_i + k1_I / 2);
    k2_I = h * f1_I(S_i + k1_S / 2, I_i + k1_I / 2);
    k2_R = h * f1_R(I_i + k1_I / 2);

    k3_S = h * f1_S(S_i + k2_S / 2, I_i + k2_I / 2);
    k3_I = h * f1_I(S_i + k2_S / 2, I_i + k2_I / 2);
    k3_R = h * f1_R(I_i + k2_I / 2);
```

```
    k4_S = h * f1_S(S_i + k3_S, I_i + k3_I);
    k4_I = h * f1_I(S_i + k3_S, I_i + k3_I);
    k4_R = h * f1_R(I_i + k3_I);

    % Update Values
    S(i+1) = S_i + (k1_S + 2*k2_S + 2*k3_S + k4_S) / 6;
    I(i+1) = I_i + (k1_I + 2*k2_I + 2*k3_I + k4_I) / 6;
    R(i+1) = R_i + (k1_R + 2*k2_R + 2*k3_R + k4_R) / 6;
end

% Linear Least Squares Analysis
lnI = log(I + eps); % Avoid log(0)
A = [t', ones(length(t), 1)]; % Linear system matrix
coeffs = A \ lnI'; % Least squares solution

k = coeffs(1); % Slope
lnI0 = coeffs(2); % Intercept

% Estimate beta and I(0)
I0_est = exp(lnI0);
beta_est = (k + gamma) * N / S0;

% Display Results
fprintf('Estimated I0: %.2f\n', I0_est);
fprintf('Estimated beta: %.2f\n', beta_est);

% Using only the first 10 days of data for comparison
t_short = t(1:11);
lnI_short = lnI(1:11);
A_short = [t_short', ones(length(t_short), 1)];
coeffs_short = A_short \ lnI_short';

k_short = coeffs_short(1);
lnI0_short = coeffs_short(2);
I0_est_short = exp(lnI0_short);
beta_est_short = (k_short + gamma) * N / S0;

fprintf('Using 10 days of data:\n');
fprintf('Estimated I0: %.2f\n', I0_est_short);
fprintf('Estimated beta: %.2f\n', beta_est_short);

% Plot the original data and least squares fit
figure;
plot(t, lnI, 'ro', 'DisplayName', 'ln(I) True Data');
hold on;
plot(t, A * coeffs, 'b-', 'DisplayName', 'Least Squares Fit');
xlabel('Time (days)');
ylabel('ln(I)');
legend('Location', 'best');
grid on;
title('Linear Least Squares Fit');

% Discussion
fprintf(['Discussion: Using 10 days of data provides better estimates
```

```
because it captures where the linear model is valid. ' ...
    '\n Extendihng the data to 30 days includes periods where the non-linear
model causes deviations from exponential growth. ' ...
    '\n This makes the estimates less accurate. Using 10 days of data aligns
better with the true behavior.'])
```

```
Estimated I0: 16.67
Estimated beta: 0.22
Using 10 days of data:
Estimated I0: 10.16
Estimated beta: 0.29
Discussion: Using 10 days of data provides better estimates because it
captures where the linear model is valid.
 Extendihng the data to 30 days includes periods where the non-linear model
causes deviations from exponential growth.
 This makes the estimates less accurate. Using 10 days of data aligns better
with the true behavior.
```

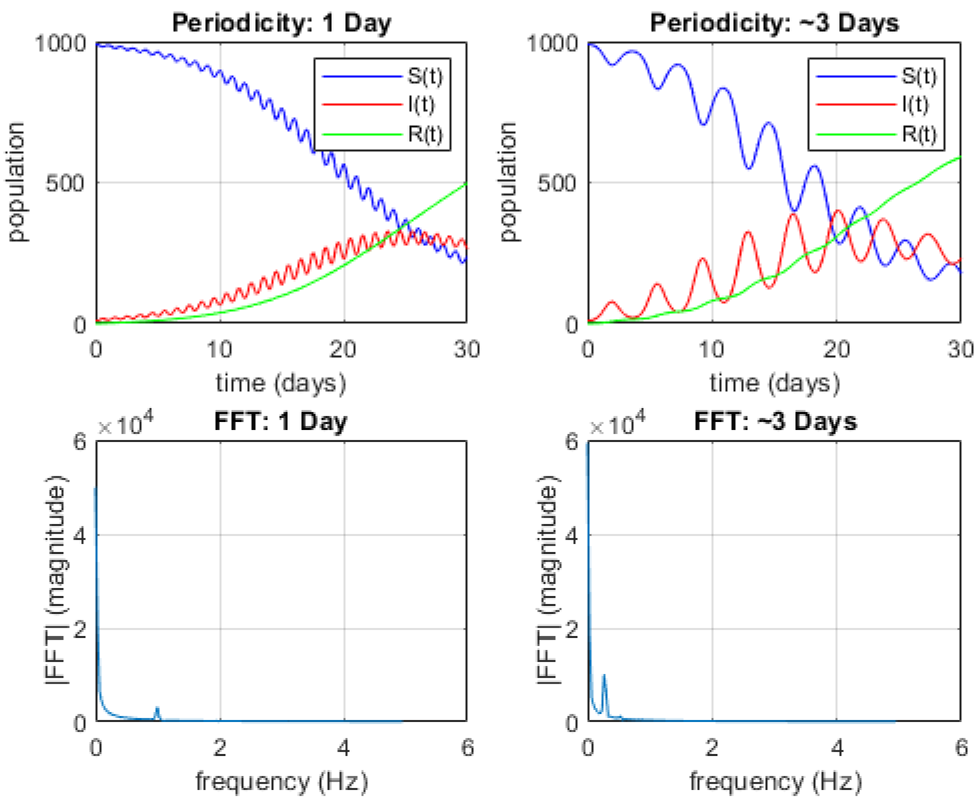


Published with MATLAB® R2024a

```
% MAE384 Project %
% Part 4%

% Parameters %

beta0 = 0.3;
A = 5;
gamma = 0.1;
S0 = 990;
I0 = 10;
R0 = 0;
h = 0.1;
T = 30;
t = 0:h:T;

% The Angular Frequencies %

omega1 = 2*pi*365/365;
omega2 = 2*pi*100/365;

%

[S1, I1, R1] = simulate_SIR_variable_beta(t,h,S0,I0,R0,beta0,A,omega1,gamma);
[S2, I2, R2] = simulate_SIR_variable_beta(t,h,S0,I0,R0,beta0,A,omega2,gamma);

[f1, I1_fft] = compute_FFT(I1, h);
[f2, I2_fft] = compute_FFT(I2, h);

% Plots for the time domains and FFT %

figure;

subplot(2, 2, 1);
plot(t, S1,'b', t, I1, 'r', t, R1, 'g');
title('Periodicity: 1 Day');
xlabel('time (days)');
ylabel('population');
legend('S(t)', 'I(t)', 'R(t)');
grid on;

subplot(2, 2, 2);
plot(t, S2, 'b', t, I2, 'r', t, R2, 'g');
title('Periodicity: ~3 Days');
xlabel('time (days)');
ylabel('population');
legend('S(t)', 'I(t)', 'R(t)');
grid on;

subplot(2, 2, 3);
plot(f1, abs(I1_fft));
title('FFT: 1 Day');
xlabel('frequency (Hz)');
```

```
ylabel('|FFT| (magnitude)');
grid on;

subplot(2, 2, 4);
plot(f2, abs(I2_fft));
title('FFT: ~3 Days');
xlabel('frequency (Hz)');
ylabel('|FFT| (magnitude)');
grid on;

% Functions %

function [S, I, R] =
simulate_SIR_variable_beta(t,h,S0,I0,R0,beta0,A,omega,gamma)

N = length(t);
S = zeros(1, N);
I = zeros(1, N);
R = zeros(1, N);
S(1) = S0;
I(1) = I0;
R(1) = R0;

for i = 1:(N-1)
    beta_t = beta0*(1 + A*sin(omega*t(i)));

    S_i = S(i);
    I_i = I(i);
    R_i = R(i);

    fS = @(S, I) -beta_t .* S .* I/(S0 + I0 + R0);
    fI = @(S, I) beta_t .* S .* I/(S0 + I0 + R0) - gamma .* I;
    fR = @(I) gamma .* I;

    k1_S = h*fS(S_i, I_i);
    k1_I = h*fI(S_i, I_i);
    k1_R = h*fR(I_i);

    k2_S = h*fS(S_i + k1_S/2, I_i + k1_I/2);
    k2_I = h*fI(S_i + k1_S/2, I_i + k1_I/2);
    k2_R = h*fR(I_i + k1_I/2);

    k3_S = h*fS(S_i + k2_S/2, I_i + k2_I/2);
    k3_I = h*fI(S_i + k2_S/2, I_i + k2_I/2);
    k3_R = h*fR(I_i + k2_I/2);

    k4_S = h*fS(S_i + k3_S, I_i + k3_I);
    k4_I = h*fI(S_i + k3_S, I_i + k3_I);
    k4_R = h*fR(I_i + k3_I);

    S(i+1) = S_i + (k1_S + 2*k2_S + 2*k3_S + k4_S) / 6;
    I(i+1) = I_i + (k1_I + 2*k2_I + 2*k3_I + k4_I) / 6;
    R(i+1) = R_i + (k1_R + 2*k2_R + 2*k3_R + k4_R) / 6;
end
```

```
end

function [frequencies, I_fft] = compute_FFT(I, h)

N = length(I);
I_fft = fft(I);

frequencies = (0:floor(N/2)-1) / (N*h);
I_fft = I_fft(1:floor(N/2));
end
```



Published with MATLAB® R2024a

Discussion for part 4:

1. Do you observe any periodic fluctuations in the signals due to periodicity of beta?

   Yes, there are periodic fluctuations in S(t), I(t), and R(t). This is because of the sinusoidal change in beta itself. When (w = 2*pi*365/365), the periodic fluctuations in I(t) should be happening every day. And when (w = 2*pi*100/365), the periodic fluctuations in I(t) should happen less often, around every three days.

2. Observe the frequency peak(s) and comment on what you see. Does it make sense physically?

   I would say yes, the peaks in the graph make sense physically because they agree with the periodicity of beta. The one-day graph represents a one-day periodicity and the FFT graph represents this with a frequency peak happening at one cycle per day. And the same goes for the three-day periodicity but with a different frequency.

3. Observe the change in the peak frequency. Does it shift to lower or higher values? Discuss your observations.

   When using the lower value of (w = 2*pi*100/365), it looks like the peak frequency shifts to lower values. The behavior shown is to be expected because a longer periodicity in beta will correspond to slower cycles. Therefore, this gives us lower values for the FFt.

MAE384-Final-Project Private

Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags Go to file t Add file Code

DanielSominsky Update README.md 34ba3e4 · now 10 Commits

MAE384 Final Project Discussion1.docx	The Discussion part for my Part1	5 days ago
MAE384 Part 2 Linear Interpolation	Update MAE384 Part 2 Linear Interpolation	yesterday
MAE384 Part 3 Least Squares	Rename MAE384_Final_LeastSquares to MAE384 Part 3 Least...	3 days ago
MAE384_Final_Project_1.m	Part 1 for MAE384 Final Project	last week
Project_Part4Graphs_MAE384.fig	Add files via upload	yesterday
Project_Part4_MAE384.m	Add files via upload	yesterday
README.md	Update README.md	now

README

MAE384-Final-Project

This Github is for our MAE384's Final Project

Please commit your individual coding parts, as well as discussions

About

No description, website, or topics provided.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors 4

DanielSominsky

jmorales48

not-rudy

Jeremiah-ASU

Languages

Files

main

Go to file t

MAE384 Final Project Discussion...
MAE384 Part 2 Linear Interpolati...
MAE384 Part 3 Least Squares
MAE384_Final_Project_1.m
Project_Part4Graphs_MAE384.fig
Project_Part4_MAE384.m
README.md

MAE384-Final-Project / README.md

DanielSominsky Update README.md 34ba3e4 · 1 minute ago History

Preview Code Blame 4 lines (3 loc) · 137 Bytes Code 55% faster with GitHub Copilot Raw

MAE384-Final-Project

This Github is for our MAE384's Final Project

Please commit your individual coding parts, as well as discussions

Contributors

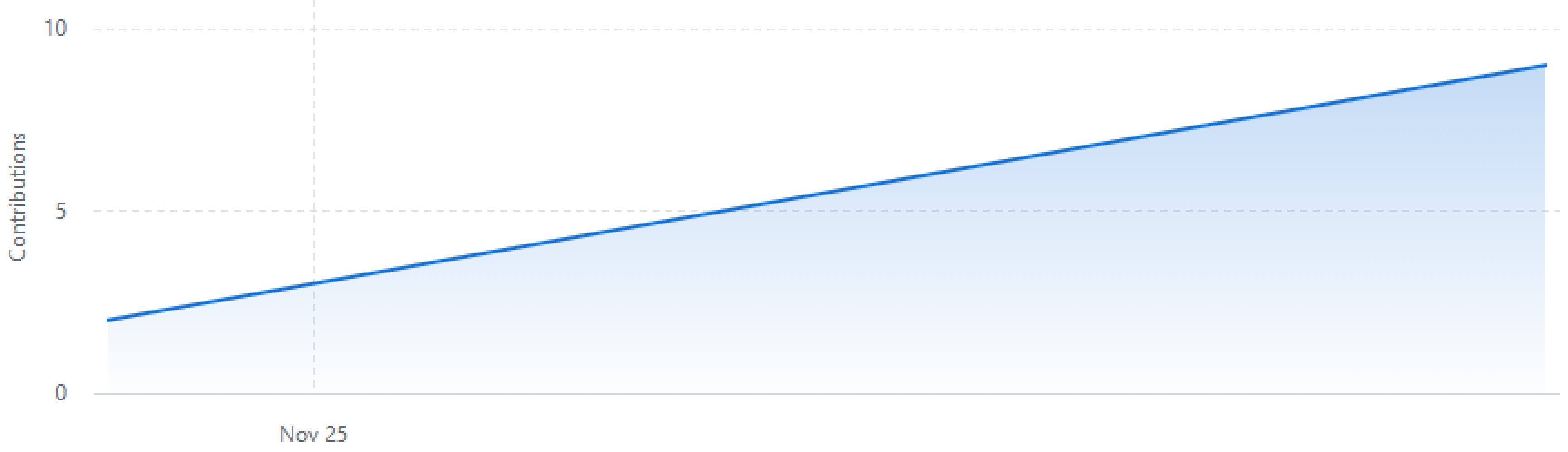
Preview Give feedback

Period: All   Contributions: Commits

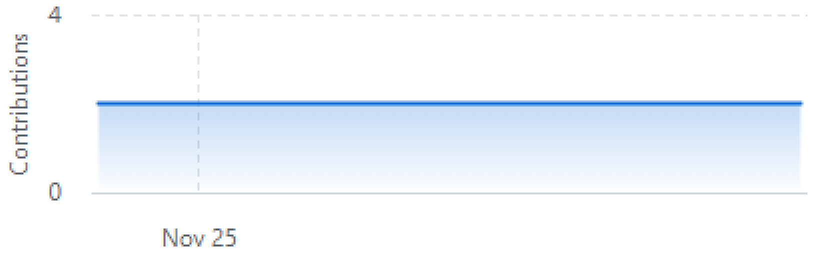
Contributions per week to main, excluding merge commits

Commits over time

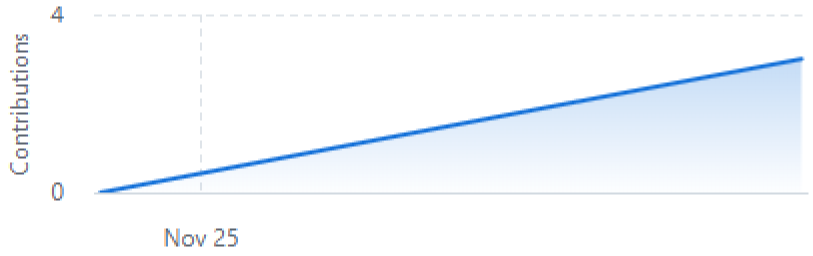
Weekly from Nov 23, 2024 to Nov 30, 2024



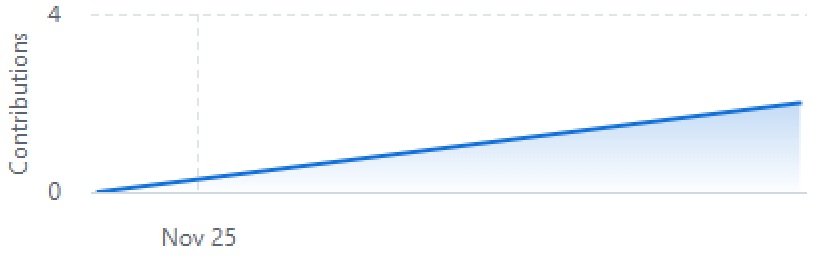
DanielSominsky #1
4 commits 85 ++ 1 --



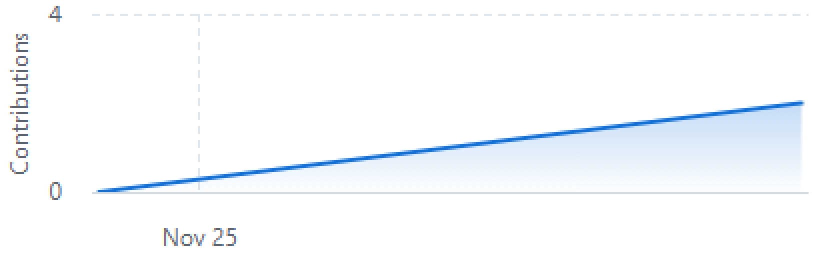
jmorales48 #2
3 commits 243 ++ 106 --



not-rudy #3
2 commits 218 ++ 109 --



Jeremiah-ASU #4
2 commits 116 ++ 0 --



DanielSominsky

Edit profile

Joined 2 weeks ago

Popular repositories

Customize your pins

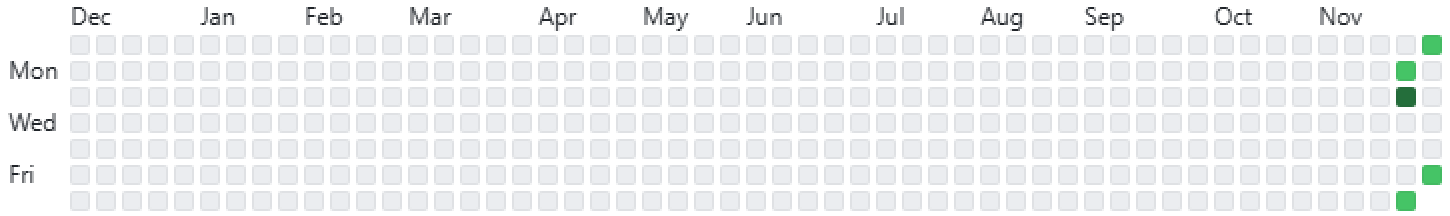
MAE384-Final-Project Public

MATLAB

6 contributions in the last year

Contribution settings

2024



Learn how we count contributions

Less More

Contribution activity

December 2024

Created 2 commits in 1 repository

DanielSominsky/MAE384-Final-Project 2 commits

Show more activity

Seeing something unexpected? Take a look at the GitHub profile guide.

jmorales48

Follow

Joined last week

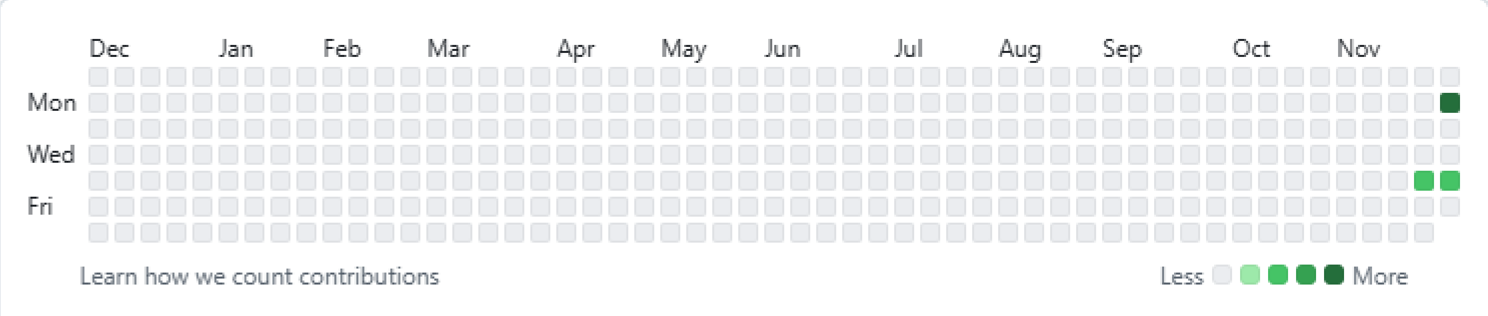
Block or Report

Popular repositories

jmorales48 doesn't have any public repositories yet.

4 contributions in the last year

2024



Contribution activity

December 2024

Created 3 commits in 1 repository

DanielSominsky/MAE384-Final-Project 3 commits

Show more activity

Seeing something unexpected? Take a look at the GitHub profile guide.

not-rudy

Follow

Joined 4 days ago

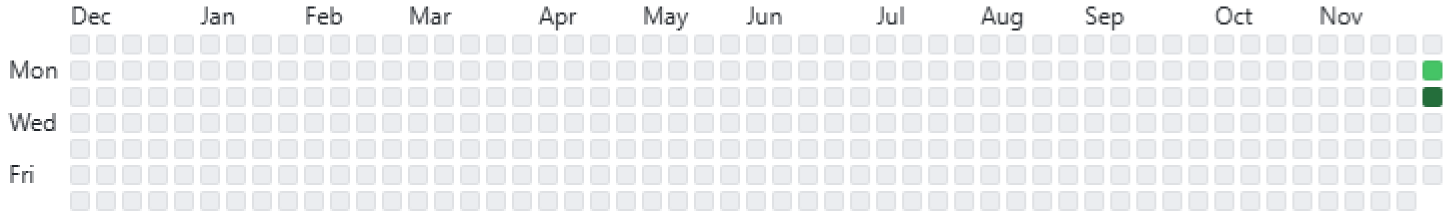
Block or Report

Popular repositories

not-rudy doesn't have any public repositories yet.

3 contributions in the last year

2024



Learn how we count contributions

Less More

Contribution activity

December 2024

Created 2 commits in 1 repository

DanielSominsky/MAE384-Final-Project 2 commits

Jeremiah-ASU

Follow

Joined 3 days ago

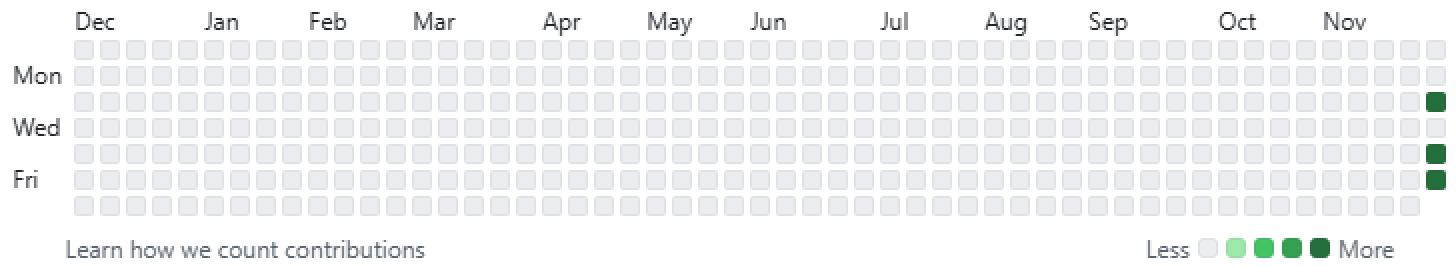
Block or Report

Popular repositories

Jeremiah-ASU doesn't have any public repositories yet.

3 contributions in the last year

2024



Learn how we count contributions

Contribution activity

December 2024

Created 2 commits in 1 repository

DanielSominsky/MAE384-Final-Project 2 commits