Table of Contents


```
load("COVID_STL.mat");

n = 6; %variable for dimensionality of A
```

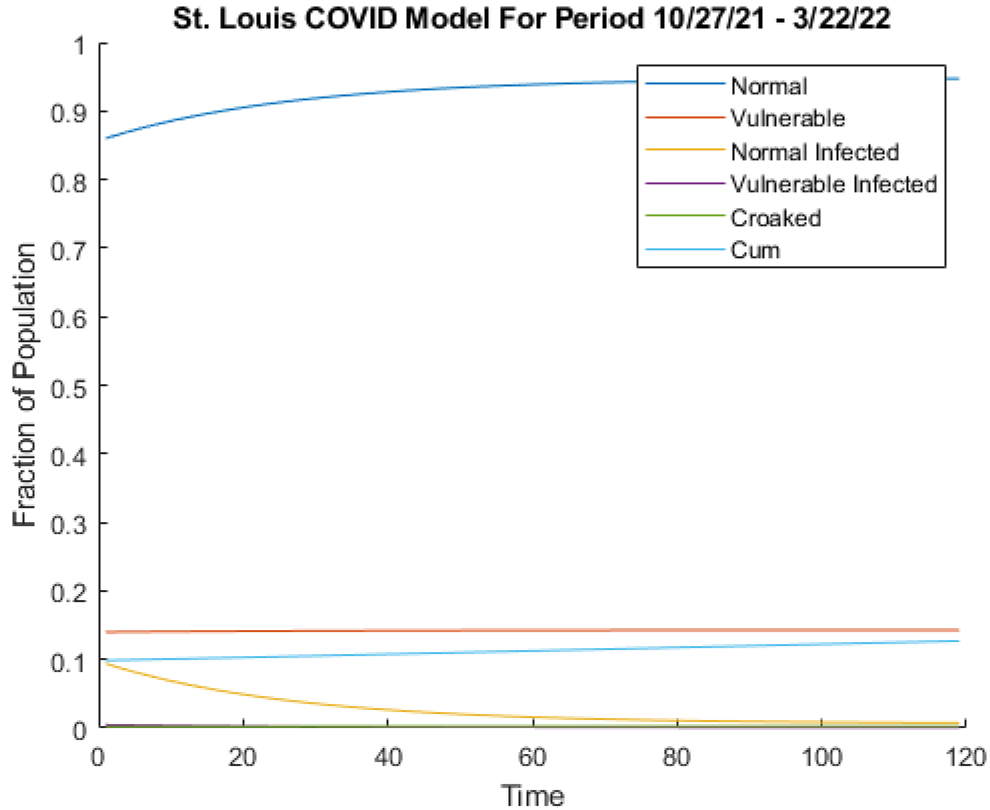
Begin delta date range

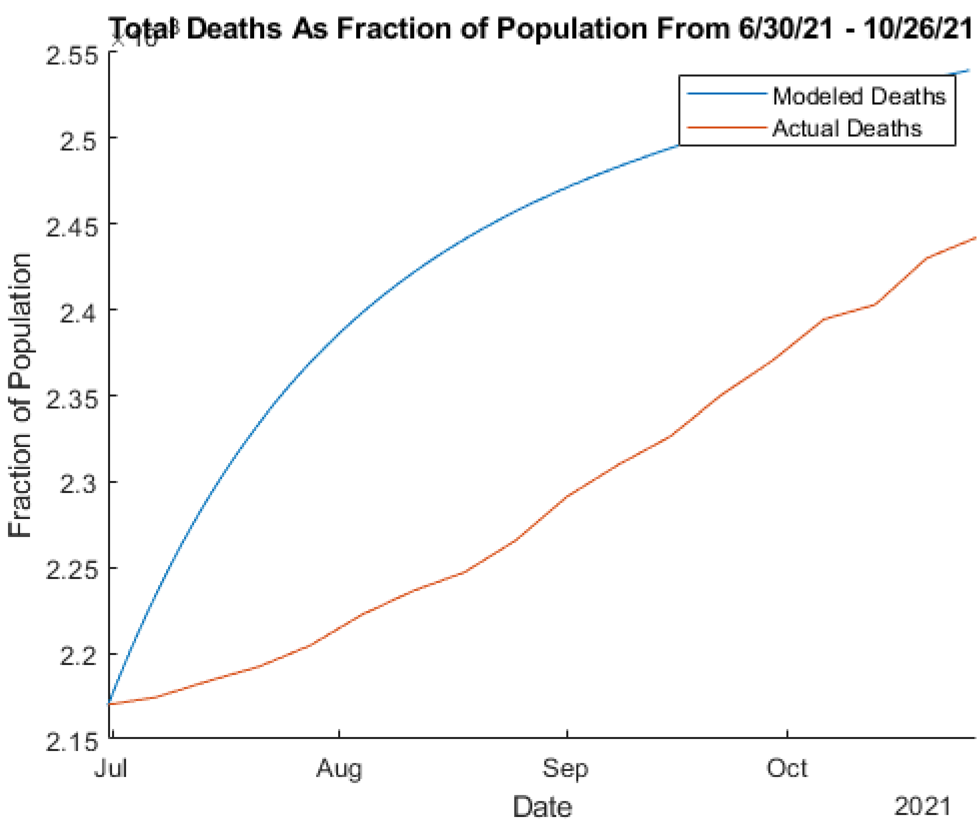
```
A = [
    0.999750 0.000000 0.037000 0.000000 0.000 0.000;
    0.000000 0.999938 0.000000 0.020000 0.000 0.000;
    0.000250 0.000000 0.962900 0.005015 0.000 0.000;
    0.000000 0.000062 0.000000 0.974700 0.000 0.000;
    0.000000 0.000000 0.000100 0.000300 1.000 0.000; %death
    0.000250 0.000062 0.000000 0.000000 0.000 1.000;
];

B = zeros(n,1);
percentAtRisk = 0.14;
percentNormal = 1 - percentAtRisk;


dailyDates = linspace(dates(1),dates(end),length(dates)*7); %create 158*7
 daily dates spanning the range of virus propagation
startDateIndex = 473; %index of the first date of the rnage
endDateIndex = 592; %index of the last date of the range
weekIndexSTART = round(startDateIndex / 7); %for indexing into dates
weekIndexEND = round(endDateIndex /7 ); %for indexing into dates
startDate = dailyDates(startDateIndex); %get datetime formatted start date
endDate = dailyDates(endDateIndex); %get the datetime formatted last date
d = endDateIndex - startDateIndex; %number of days to simulate for, equal to
 the final index of the date range, or the date number
startingNormalInfected = cases_STL(weekIndexSTART) * 0.9533; %normal cases are
 the rest of the non-vulnerable cases
startingDeaths = deaths_STL(weekIndexSTART);
startingVulnerableInfected = cases_STL(weekIndexSTART) * 0.0467; %vulnerable
 cases should be 1/3 of 14% of the total population


x0 = [
    (POP_STL * percentNormal);
    (POP_STL * percentAtRisk);
    startingNormalInfected;
```

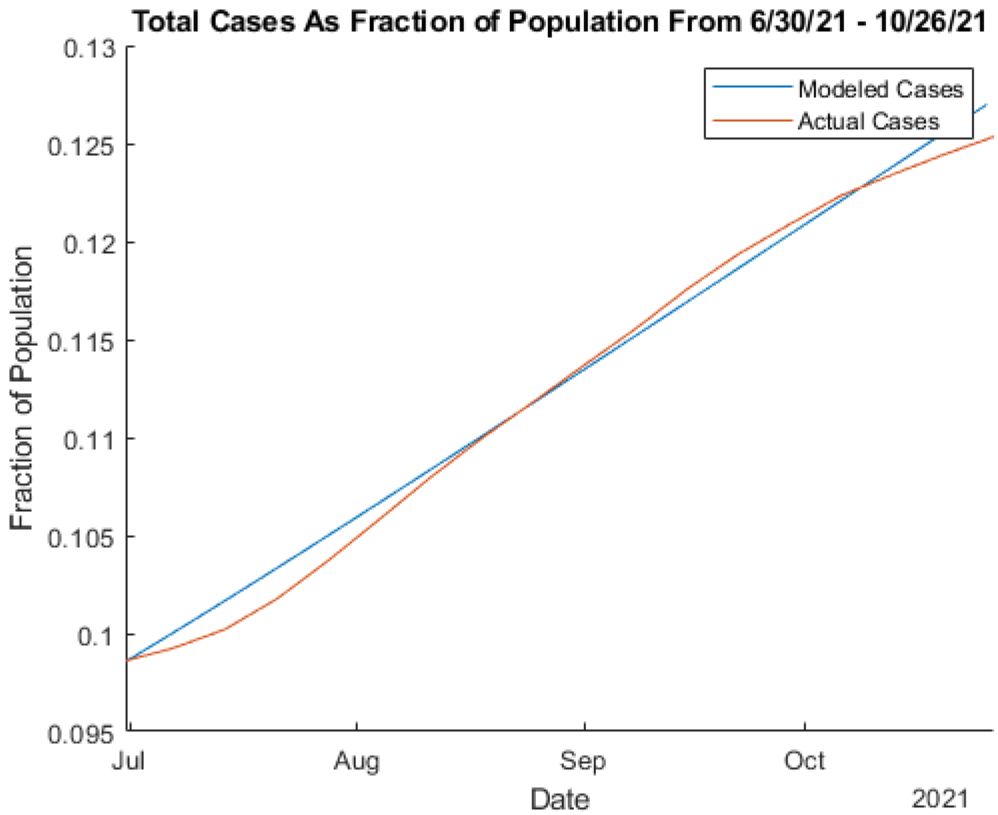
```
    startingVulnerableInfected;
    startingDeaths;
    startingVulnerableInfected + startingNormalInfected%total starting cases
 is the sum of
    % starting normal infected and starting vulnerable infected data
 ];

sys_sir_base = ss(A,B,eye(n) ,zeros(n,1),1);
Y = lsim(sys_sir_base,zeros(d,1),linspace(0,d - 1,d),x0); %simulate for d days
 of spread
origY = Y; %leave the original Y values in here
Y = Y/POP_STL; %convert SIRD values to a fraction of the whole STL population
% plot the output trajectory
figure;
hold on; %toggle hold, plotting multiple curves on the same graph
plot(Y(1:d,1:n));
legend('Normal', 'Vulnerable','Normal Infected','Vulnerable
 Infected','Croaked','Cum');
title('St. Louis COVID Model For Period 10/27/21 - 3/22/22')
xlabel('Time')
ylabel('Fraction of Population');
ylim auto; hold off;


%casesFraction = cases_STL / POP_STL; %create new case vector storing cases as
 fraction of whole population
%plot(casesFraction(1:100));

figure;
hold on;
plot(dailyDates(startDateIndex:endDateIndex - 1),origY(1:d,5) / POP_STL);
plot(dates(weekIndexSTART:weekIndexEND),deaths_STL(weekIndexSTART:weekIndexEND)/
POP_STL);
xlim([startDate endDate]);
title('Total Deaths As Fraction of Population From 6/30/21 - 10/26/21');
legend('Modeled Deaths','Actual Deaths');
ylabel('Fraction of Population');
xlabel('Date');

figure;
hold on;
plot(dailyDates(startDateIndex:endDateIndex -1),origY(1:d,n) /
 POP_STL); %trust me it works
plot(dates(weekIndexSTART:weekIndexEND),cases_STL(weekIndexSTART:weekIndexEND) /
 POP_STL); %need to build this such taht it is same length as number of days
 that we want to store so we can plot them together
legend('Modeled Cases', 'Actual Cases');
xlim([startDate endDate]);
title('Total Cases As Fraction of Population From 6/30/21 - 10/26/21');
ylabel('Fraction of Population');
xlabel('Date');
hold off;
```

```
casesError = 0;
samples = 0;
funnyWeekIndex = weekIndexSTART; %we need 2 of these for each error
 calculation, this one gonna get incremented
for i = 1:7:d %below is used for calculating error between model and actual
    samples = samples + 1; %increment samples used to track number of tests,
 important bc working w/ multiples of 7
    %we can also use the above count variable to access weekly entries in
    %cases_STL
    modeledCases = origY(i,6); %access a point from each week, reported on the
 same day as the actual data
    actualCases = cases_STL(funnyWeekIndex); %cases STL contains weekly data
    tempError = ((modeledCases - actualCases) / actualCases) * 100; %calculate
 weekly error
    casesError = casesError + tempError;
    funnyWeekIndex = funnyWeekIndex + 1; %for indexing into weekly data
end
casesError = casesError/samples;
fprintf('First Range Cases Average Percent Error: %.2f%%\n', casesError);


deathsError = 0;
samples = 0;
funnyWeekIndex = weekIndexSTART;
for i = 1:7:d %below is used for calculating error between model and actual
    samples = samples + 1; %increment samples used to track number of tests,
 important bc working w/ multiples of 7
    %we can also use the above count variable to access weekly entries in
    %cases_STL
    modeledDeaths = origY(i,5); %access a point from each week, reported on
 the same day as the actual data
    actualDeaths = deaths_STL(funnyWeekIndex); %deaths STL contains weekly
 data
    tempError = ((modeledDeaths - actualDeaths) / actualDeaths) *
 100; %calculate weekly error
    deathsError = deathsError + tempError;
    funnyWeekIndex = funnyWeekIndex + 1; %for indexing into weekly data
end

deathsError = deathsError/samples;
fprintf('First Range Deaths Average Percent Error: %.2f%%\n', deathsError);


%-----------------------------------------------------------------
```

First Range Cases Average Percent Error: 0.33%
First Range Deaths Average Percent Error: 6.18%


St. Louis COVID Model For Period 10/27/21 - 3/22/22
Normal
Vulnerable
Normal Infected
Vulnerable Infected
Croaked
Cum
Fraction of Population
Time
0
20
40
60
80
100
120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Total Deaths As Fraction of Population From 6/30/21 - 10/26/21
Modeled Deaths
Actual Deaths
Fraction of Population
Date
Jul
Aug
Sep
Oct
2021
2.15
2.2
2.25
2.3
2.35
2.4
2.45
2.5
2.55



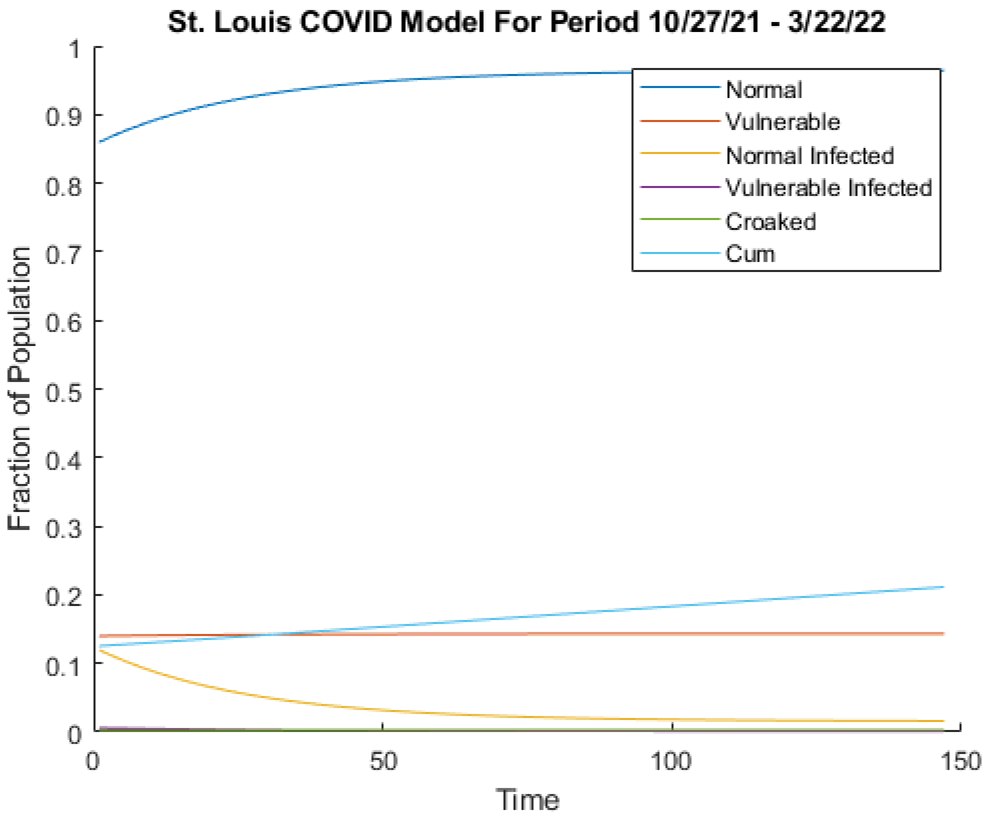
begin omicron data range

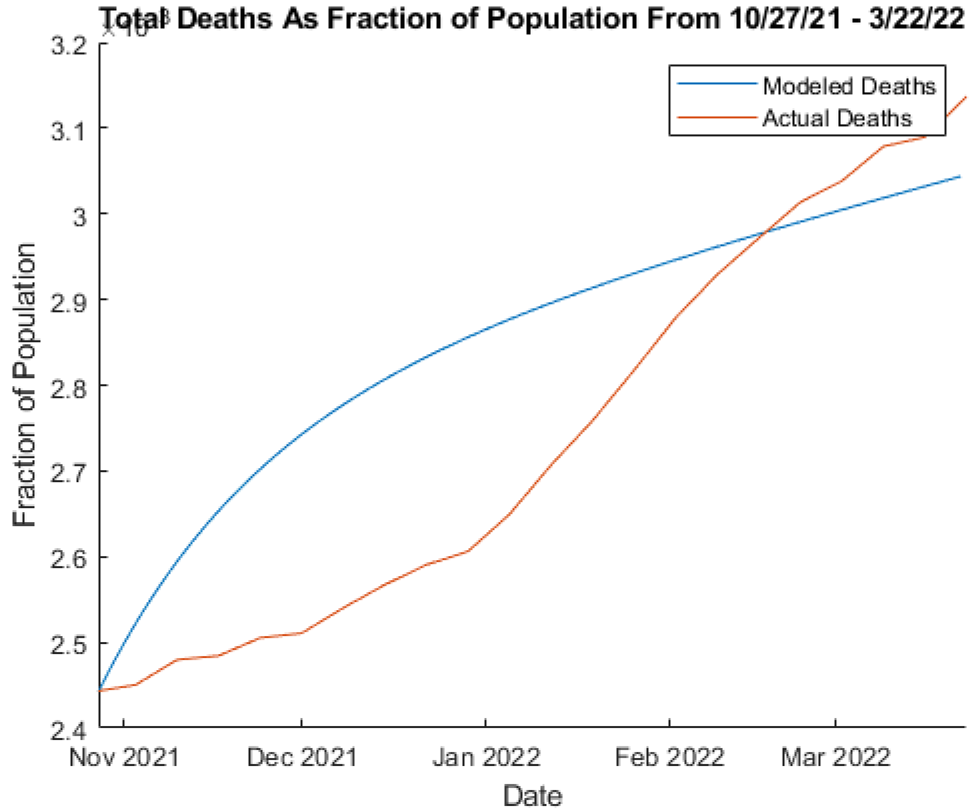
```
%----------------------------------------------------------------

A = [ %begin by rebuilding A to reflect changes in case trajectories in
 10/27/21 - 3/22/22 time range
    0.999400 0.000000 0.037000 0.000000 0.000 0.000;
    0.000000 0.999850 0.000000 0.020000 0.000 0.000;
    0.000600 0.000000 0.962900 0.005015 0.000 0.000;
    0.000000 0.000150 0.000000 0.974700 0.000 0.000;
    0.000000 0.000000 0.000100 0.000300 1.000 0.000; %death
    0.000600 0.000150 0.000000 0.000000 0.000 1.000;
];


startDateIndex = 593; %index of the first date of the rnage
endDateIndex = 740; %index of the last date of the range
weekIndexSTART = round(startDateIndex / 7); %for indexing into dates
weekIndexEND = round(endDateIndex / 7); %for indexing into dates
startDate = dailyDates(startDateIndex); %get datetime formatted start date
endDate = dailyDates(endDateIndex); %get the datetime formatted last date
d = endDateIndex - startDateIndex; %number of days to simulate for, equal to
 the final index of the date range, or the date number
```

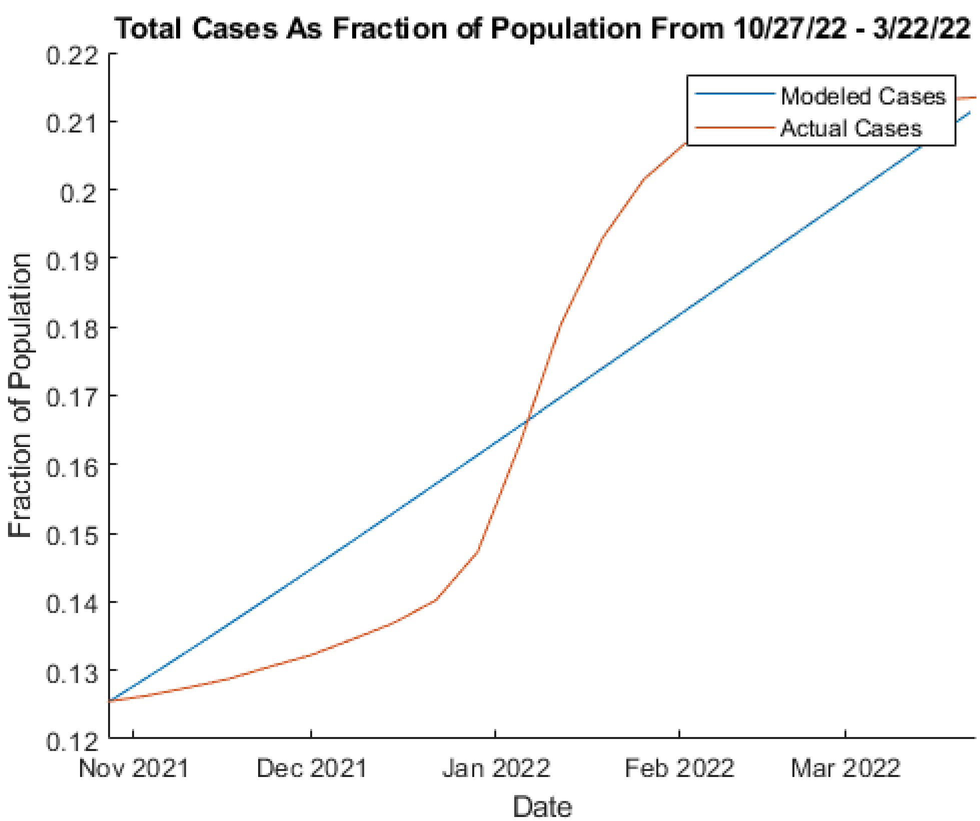
```
startingNormalInfected = cases_STL(weekIndexSTART) * 0.9533; %normal cases are
 the rest of the non-vulnerable cases
startingDeaths = deaths_STL(weekIndexSTART);
startingVulnerableInfected = cases_STL(weekIndexSTART) * 0.0467; %vulnerable
 cases should be 1/3 of 14% of the total population

x0 = [
    (POP_STL * percentNormal);
    (POP_STL * percentAtRisk);
    startingNormalInfected;
    startingVulnerableInfected;
    startingDeaths;
    startingVulnerableInfected + startingNormalInfected%total starting cases
 is the sum of
    % starting normal infected and starting vulnerable infected data
 ];

sys_sir_base = ss(A,B,eye(n) ,zeros(n,1),1);
Y = lsim(sys_sir_base,zeros(d,1),linspace(0,d - 1,d),x0); %simulate for d days
 of spread
origY = Y; %leave the original Y values in here
Y = Y/POP_STL; %convert SIRD values to a fraction of the whole STL population
% plot the output trajectory
figure;
hold on; %toggle hold, plotting multiple curves on the same graph
plot(Y(1:d,1:n));
legend('Normal', 'Vulnerable','Normal Infected','Vulnerable
 Infected','Croaked','Cum');
title('St. Louis COVID Model For Period 10/27/21 - 3/22/22')
xlabel('Time')
ylabel('Fraction of Population');
ylim auto; hold off;

%casesFraction = cases_STL / POP_STL; %create new case vector storing cases as
 fraction of whole population
%plot(casesFraction(1:100));

figure;
hold on;
plot(dailyDates(startDateIndex:endDateIndex - 1),origY(1:d,5) / POP_STL);
plot(dates(weekIndexSTART:weekIndexEND),deaths_STL(weekIndexSTART:weekIndexEND)/
POP_STL);
xlim([startDate endDate]);
title('Total Deaths As Fraction of Population From 10/27/21 - 3/22/22');
legend('Modeled Deaths','Actual Deaths');
ylabel('Fraction of Population');
xlabel('Date');

figure;
hold on;
plot(dailyDates(startDateIndex:endDateIndex -1),origY(1:d,n) /
 POP_STL); %trust me it works
```

```
plot(dates(weekIndexSTART:weekIndexEND),cases_STL(weekIndexSTART:weekIndexEND) /
 POP_STL); %need to build this such taht it is same length as number of days
 that we want to store so we can plot them together
legend('Modeled Cases', 'Actual Cases');
xlim([startDate endDate]);
title('Total Cases As Fraction of Population From 10/27/22 -
 3/22/22');ylabel('Fraction of Population');
ylabel('Fraction of Population');
xlabel('Date');
hold off;



casesError = 0;
samples = 0;
funnyWeekIndex = weekIndexSTART; %we need 2 of these for each error
 calculation, this one gonna get incremented
for i = 1:7:d %below is used for calculating error between model and actual
    samples = samples + 1; %increment samples used to track number of tests,
 important bc working w/ multiples of 7
    %we can also use the above count variable to access weekly entries in
    %cases_STL
    modeledCases = origY(i,6); %access a point from each week, reported on the
 same day as the actual data
    actualCases = cases_STL(funnyWeekIndex); %cases STL contains weekly data
    tempError = ((modeledCases - actualCases) / actualCases) * 100; %calculate
 weekly error
    casesError = casesError + tempError;
    funnyWeekIndex = funnyWeekIndex + 1; %for indexing into weekly data
end
casesError = casesError/samples;
fprintf('Second Range Cases Average Percent Error: %.2f%%\n', casesError);


deathsError = 0;
samples = 0;
funnyWeekIndex = weekIndexSTART;
for i = 1:7:d %below is used for calculating error between model and actual
    samples = samples + 1; %increment samples used to track number of tests,
 important bc working w/ multiples of 7
    %we can also use the above count variable to access weekly entries in
    %cases_STL
    modeledDeaths = origY(i,5); %access a point from each week, reported on
 the same day as the actual data
    actualDeaths = deaths_STL(funnyWeekIndex); %deaths STL contains weekly
 data
    tempError = ((modeledDeaths - actualDeaths) / actualDeaths) *
 100; %calculate weekly error
    deathsError = deathsError + tempError;
    funnyWeekIndex = funnyWeekIndex + 1; %for indexing into weekly data
end

deathsError = deathsError/samples;
fprintf('Second Range Deaths Average Percent Error: %.2f%%\n', deathsError);
```

```
%---------------------------------------------------------------------------

Second Range Cases Average Percent Error: -0.06%
Second Range Deaths Average Percent Error: 4.47%
```



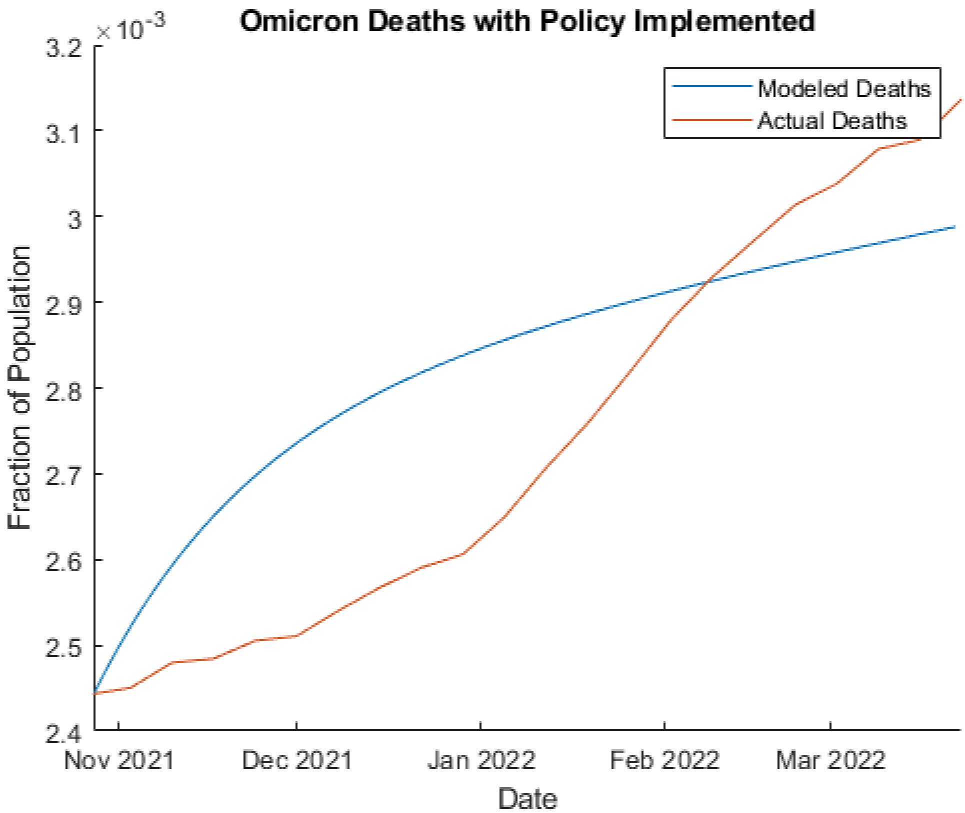

Total Deaths As Fraction of Population From 10/27/21 - 3/22/22
Modeled Deaths
Actual Deaths
Fraction of Population
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
Nov 2021
Dec 2021
Jan 2022
Feb 2022
Mar 2022
Date

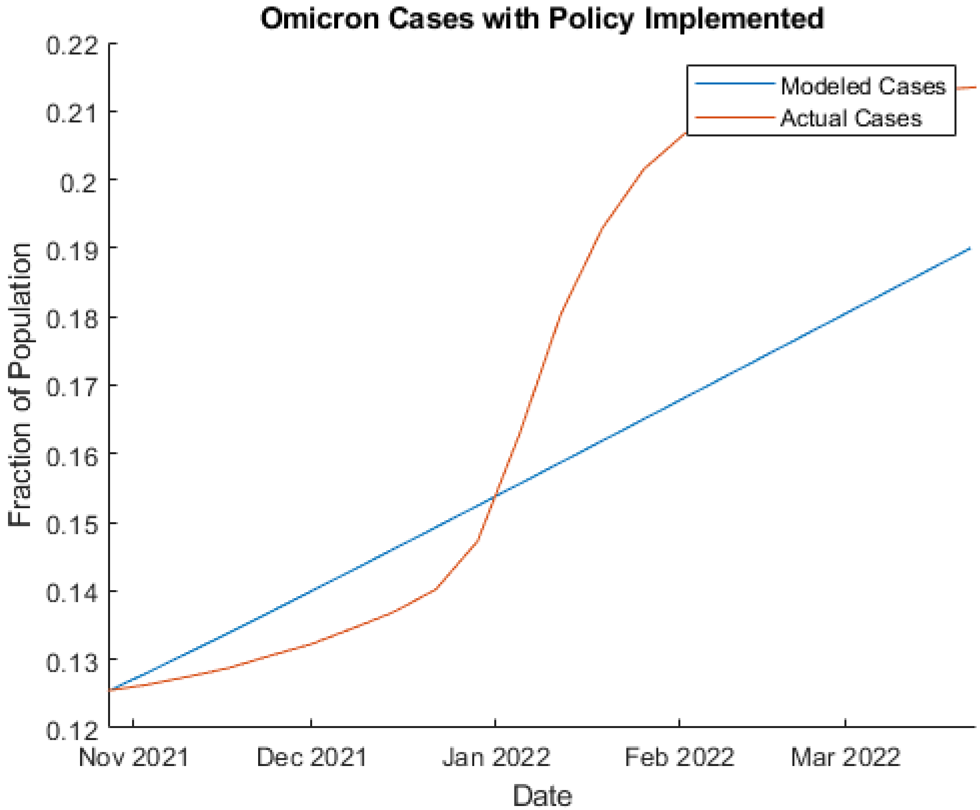

Total Cases As Fraction of Population From 10/27/22 - 3/22/22
Modeled Cases
Actual Cases
Fraction of Population
0.22
0.21
0.2
0.19
0.18
0.17
0.16
0.15
0.14
0.13
0.12
Nov 2021
Dec 2021
Jan 2022
Feb 2022
Mar 2022
Date

begin policy analysis

```
%--------------------------------------------------------------------
A = [ %reduced A matrix using values for the omicron period. Infections and
 Deaths are 25% less common
    %than in the original model w/ 0.06% error compared to actual case data
    %this is accomplished by simply reducing infection rate by 25%, as 25%
    %less people will die as a result of this change so no need to mess
    %with the numbers for deaths as well

    0.999550 0.000000 0.037000 0.000000 0.000 0.000;
    0.000000 0.999888 0.000000 0.025015 0.000 0.000;
    0.000450 0.000000 0.962900 0.000000 0.000 0.000;
    0.000000 0.000112 0.000000 0.974700 0.000 0.000;
    0.000000 0.000000 0.000100 0.000300 1.000 0.000; %death row. haha get it
    0.000450 0.000112 0.000000 0.000000 0.000 1.000;
];

startDateIndex = 593; %index of the first date of the rnage
endDateIndex = 740; %index of the last date of the range
weekIndexSTART = round(startDateIndex / 7); %for indexing into dates
weekIndexEND = round(endDateIndex / 7); %for indexing into dates
startDate = dailyDates(startDateIndex); %get datetime formatted start date
endDate = dailyDates(endDateIndex); %get the datetime formatted last date
d = endDateIndex - startDateIndex; %number of days to simulate for, equal to
 the final index of the date range, or the date number
startingNormalInfected = cases_STL(weekIndexSTART) * 0.9533; %normal cases are
 the rest of the non-vulnerable cases
startingDeaths = deaths_STL(weekIndexSTART);
startingVulnerableInfected = cases_STL(weekIndexSTART) * 0.0467; %vulnerable
 cases should be 1/3 of 14% of the total population

x0 = [
    (POP_STL * percentNormal);
    (POP_STL * percentAtRisk);
    startingNormalInfected;
    startingVulnerableInfected;
    startingDeaths;
    startingVulnerableInfected + startingNormalInfected%total starting cases
 is the sum of
    % starting normal infected and starting vulnerable infected data
 ];

sys_sir_base = ss(A,B,eye(n) ,zeros(n,1),1);
Y = lsim(sys_sir_base,zeros(d,1),linspace(0,d - 1,d),x0); %simulate for d days
 of spread
origY = Y; %leave the original Y values in here
Y = Y/POP_STL; %convert SIRD values to a fraction of the whole STL population
% plot the output trajectory


figure;
hold on;
```

```
plot(dailyDates(startDateIndex:endDateIndex - 1),origY(1:d,5) / POP_STL);
plot(dates(weekIndexSTART:weekIndexEND),deaths_STL(weekIndexSTART:weekIndexEND)/
POP_STL);
xlim([startDate endDate]);
title('Omicron Deaths with Policy Implemented');
legend('Modeled Deaths','Actual Deaths');
ylabel('Fraction of Population');
xlabel('Date');

figure;
hold on;
plot(dailyDates(startDateIndex:endDateIndex -1),origY(1:d,n) /
 POP_STL); %trust me it works
plot(dates(weekIndexSTART:weekIndexEND),cases_STL(weekIndexSTART:weekIndexEND) /
 POP_STL); %need to build this such taht it is same length as number of days
 that we want to store so we can plot them together
legend('Modeled Cases', 'Actual Cases');
xlim([startDate endDate]);
title('Omicron Cases with Policy Implemented');
ylabel('Fraction of Population');
xlabel('Date');
hold off;
```





Policy Design Questions

```
%(a) What is your policy? How is it implemented mathematically in the model?
 Does it achieve the desired effect?
%Our policy is a mask mandate that requires all individuals in St. Louis to
%wear a protective face covering when in public. The impact of this policy
%is represented mathematically in our lower infection rates (decreased by
%25% from the standard model). As a result of this change, 25% less normal
%and 25% less vulnerable individuals become infected with Covid in our
%updated model. The policy does achieve the desired effect, as our plots of
%the updated model show considerably less cases and deaths compared to the
%actual data given to us. Less deaths and less infections is the goal of
%the policy, so this marks it as successful.
%(b) Is your policy feasible? In other words, will the societal costs be too
 great for this policy to bevworthwhile?
%Our policy is almost certainly feasible. By way of mandating masks, we
%prevent thousands of infections and save hundreds of lives. The impact of
%this is multifaceted. By preventing infections, St. Louis is more
%economically productive. However, saving lives is obviously paramount and
%is something that is accomplished with a relatively simple task as we
%demonstrate in our updated model, marking this policy as worthwhile when
%compared to its minimal social cost.
```

Published with MATLAB® R2023a