HW4

Preprocessing

```
clear global

% Reading in the Excel file as a table then converting it into a matrix
mytable = readtable('project8_data.xlsx');
x = mytable{2:3,13:end};
x = transpose(x);

% 1st column: cumulative number of detected infections
V = x(:,1)
```

```
V = 1091×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

```
% 2nd column: Covid-19 related deaths reported in that specific county/city
Y = x(:,2)
```

```
Y = 1091×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

Exercise One Preprocessing

```
%%
Nmax = 236842; % Max population; from Population column
Tmax = 119;    % Number of days we will attempt to model
Vmin = 5;      % See below
tau0 = 7;      % Time between infection and full symptom onset
h = 0.01;      % Step size

% Sets used in omega set generation
alphaSet = 0.05:0.01:0.2;
R0Set = 1.5:0.1:1.9;
```

```
NfracSet = 0.02:0.01:0.1;
deltaSet = 0.05:0.01:0.4;

% Norm used in error calculation
pSet = [1 2 inf];
pLen = length(pSet);

% Get the first day where at least Vmin were detected as infected
for i = 1:size(x, 1)
    if x(i,1) >= 5
        break
    end
end
t0 = i
```

```
t0 = 52
```

```
% Preprocess rate of infections
I = zeros(Tmax+1,1); % note that I(t) represent the value of I at t+1
for t=0:Tmax
    I(t+1) = V(t+t0+tau0) - V(t+t0-tau0);
end
I;

I0 = I(1)
```

```
I0 = 17
```

Exercise One Part 1

```
% Get parameters
% Function defined in generateParams1.m
[alphaLen, betaLen, NLen, omega1] = ...
    generateParams1(alphaSet, R0Set, NfracSet, Nmax);

% Testing Euler scheme function defined in SEIR_euler.m
params = num2cell(squeeze(omega1(1,1,1,:)));
[alpha, beta, N] = params{:}
```

```
alpha = 0.0500
beta = 0.0750
N = 4.7368e+03
```

```
[Ssim, Isim, Rsim] = SIR_euler(I0,Tmax,alpha,beta,N)
```

Ssim = 120×1

10^3 ×

```
    4.7368
    4.7355
    4.7342
    4.7329
    4.7315
    4.7301
    4.7286
    4.7271
```

```
    4.7256
    4.7240
      ⋮
Isim = 120×1
   17.0000
   17.4301
   17.8708
   18.3222
   18.7845
   19.2582
   19.7433
   20.2401
   20.7490
   21.2702
      ⋮
Rsim = 120×1
         0
    0.8606
    1.7430
    2.6476
    3.5751
    4.5260
    5.5009
    6.5003
    7.5249
    8.5752
      ⋮
```

```
[gamma, minVal] = minimizeGamma(t0, Tmax, Y, Rsim, 1)
```

```
gamma = 0.3899
minVal = 2.1039e+03
```

```
% gammas(alphaInd, betaInd, NInd, p) contains
% gamma as calculated to minimize p-norm of residual
% of actual results as compared to euler model
% with given parameters (alpha, beta, N)
% J(...) contains the minimized function value
gammas = zeros(alphaLen, betaLen, NLen, pLen);
J = zeros(alphaLen, betaLen, NLen, pLen);

fprintf("alphas iterated over:")
```

```
alphas iterated over:
```

```
for alphaInd = 1:alphaLen
    fprintf("%.2f, ", alphaSet(alphaInd))
    for betaInd = 1:betaLen
        for NInd = 1:NLen
            % Get parameters from set, use Euler scheme
            params = num2cell(squeeze(omega1(alphaInd, betaInd, NInd, :)));
            [alpha, beta, N] = params{:};
            [Ssim, Isim, Rsim] = SIR_euler(I0, Tmax, alpha, beta, N);

            % For each p, find gamma minimizing p-norm & store
            for pInd = 1:pLen
```

```
                p = pSet(pInd);
                % Function defined in minimizeGamma.m
                [gamma, ~] = minimizeGamma(t0, Tmax, Y, Rsim, p);
                gammas(alphaInd, betaInd, NInd, pInd) = gamma;
                %Y(t0:t0+Tmax)
                %size(I)
                %I(t0:t0+Tmax)
                J(alphaInd, betaInd, NInd, pInd) = objectiveFunction(Y(t0:t0+Tmax), Rsim, I, Is
            end
        end
    end
end
```

```
0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.20,
```

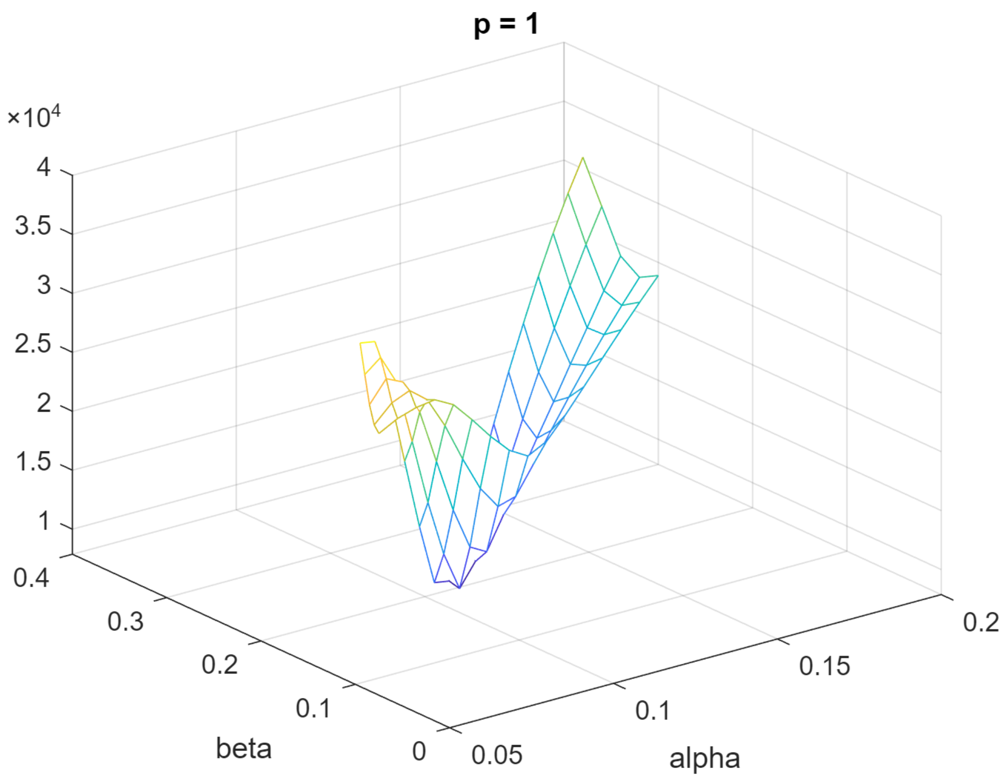
```
paramMinp = cell(pLen,4); % Store {alpha, beta, gamma, N}

% For each p value...
for pInd = 1:pLen
    p = pSet(pInd);
    Jp = J(:,:,:,pInd);
    gammasp = gammas(:,:,:,pInd);

    % Get index of minimum error
    [M, Ind] = min(Jp, [], "all");
    [alphaMinInd, betaMinInd, NMinInd] = ind2sub(size(Jp),Ind);

    % Find parameters at that index
    gammaMin = gammasp(Ind);
    omegaTemp = reshape(omega1, [], 3);
    paramMin = num2cell(omegaTemp(Ind,:));
    [alphaMin, betaMin, NMin] = paramMin{:};

    % Print result
    fprintf("For p = %d, the parameters which reduce the error are\n" + ...
        "alpha = %.3f, beta = %.3f, gamma = %.3f, N = %g\n" + ...
        "(R0 = %.3f, Nfrac = %.3f)\n" + ...
        "with an error of %f\n\n", ...
        p,alphaMin, betaMin, gammaMin, NMin, betaMin / alphaMin, NMin / Nmax, M);
    % Store minimum parameter values for Exercise 1 Part 3
    paramMinp(pInd,:) = {alphaMin, betaMin, gammaMin, NMin};
end
```

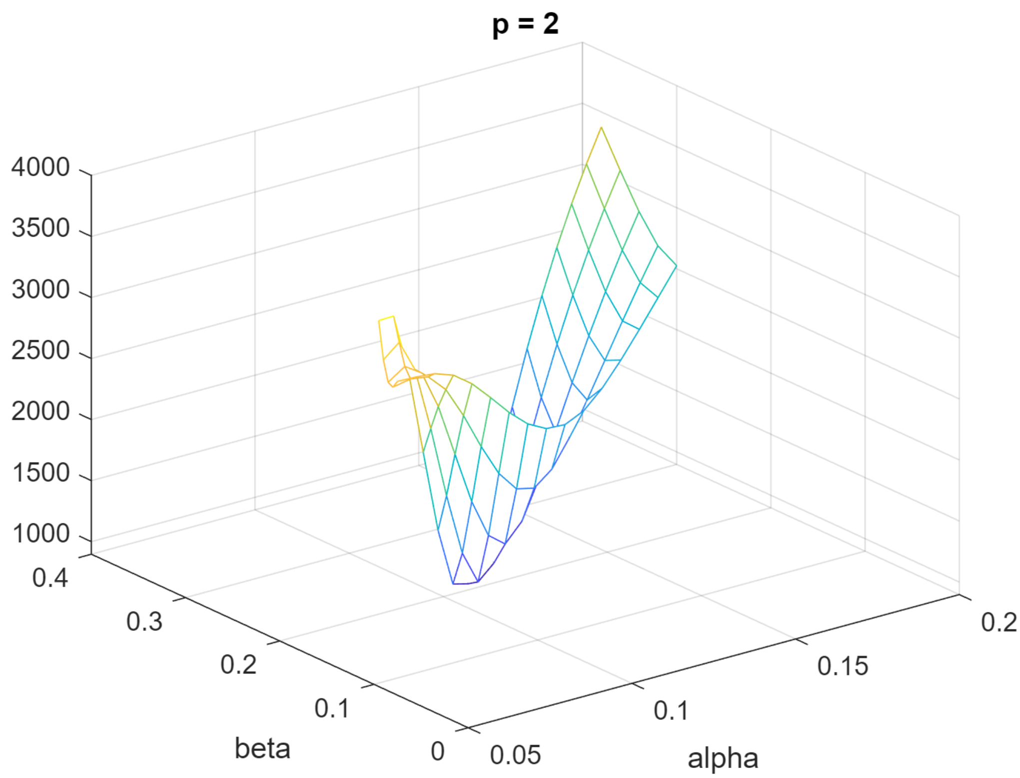
```
For p = 1, the parameters which reduce the error are
alpha = 0.110, beta = 0.198, gamma = 0.040, N = 4736.84
(R0 = 1.800, Nfrac = 0.020)
with an error of 7875.655189

For p = 2, the parameters which reduce the error are
alpha = 0.110, beta = 0.209, gamma = 0.036, N = 4736.84
(R0 = 1.900, Nfrac = 0.020)
with an error of 897.344070

For p = Inf, the parameters which reduce the error are
alpha = 0.110, beta = 0.209, gamma = 0.034, N = 4736.84
```

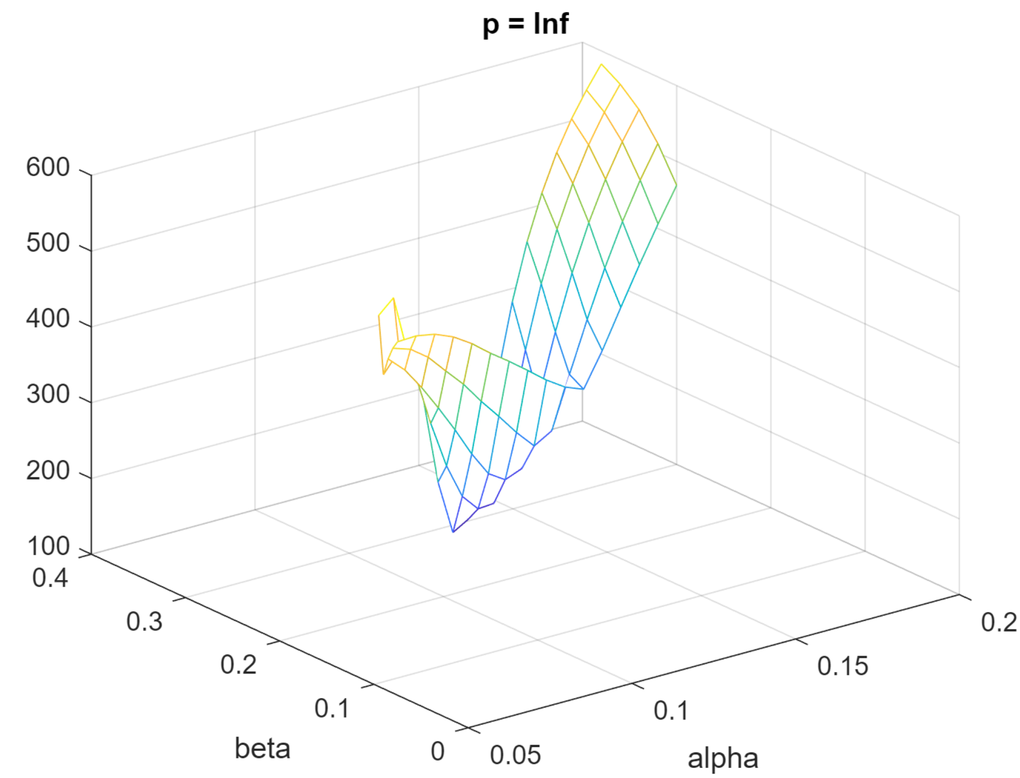
```
(R0 = 1.900, Nfrac = 0.020)
with an error of 184.585142
```

Exercise One Part 2

```
% (α, β) → J = J(α, β, N-hat, γ-hat)
for pInd = 1:pLen
    params = omega1(:, :, NMinInd, :);
    alphas = reshape(params(:,:,:,1), alphaLen, []);
    betas = reshape(params(:,:,:,2), [], betaLen);
    Jparams = J(:, :, NMinInd, pInd);

    figure
    mesh(alphas, betas, Jparams);
    title(['p = ', num2str(pSet(pInd))])
    xlabel("alpha"); ylabel("beta")
end
```



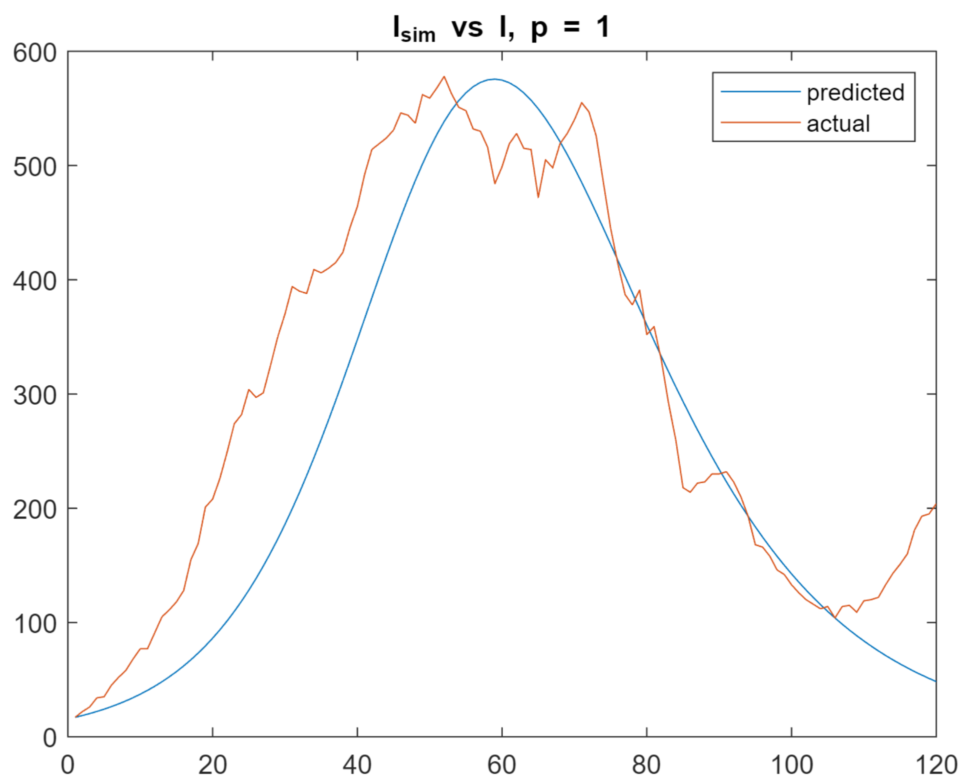


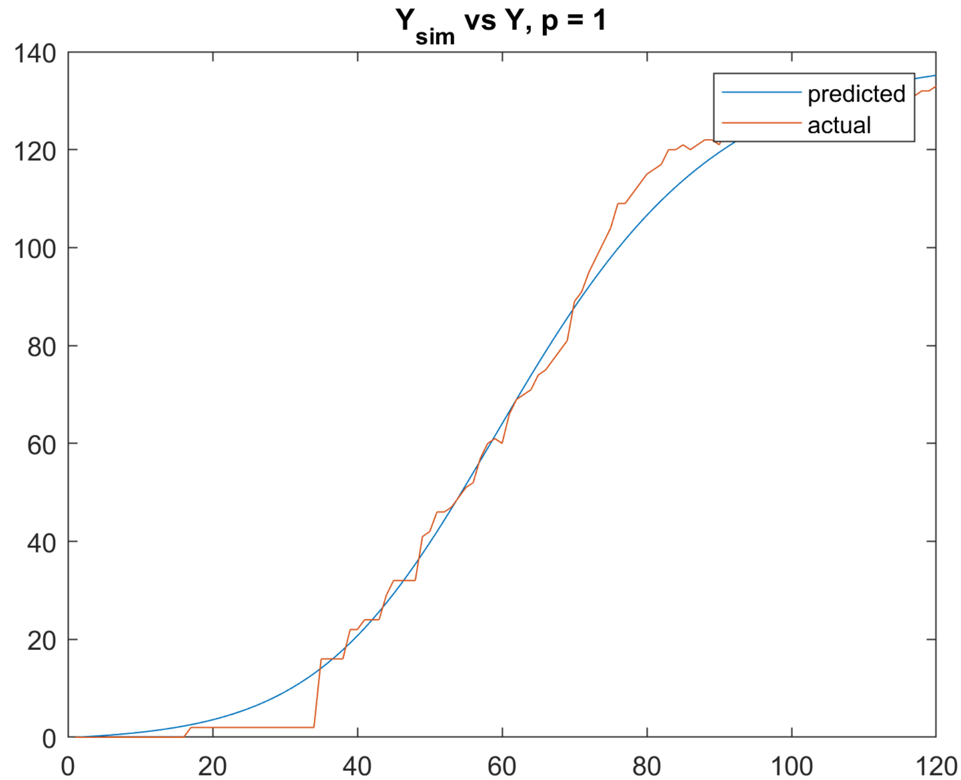


Exercise One Part 3

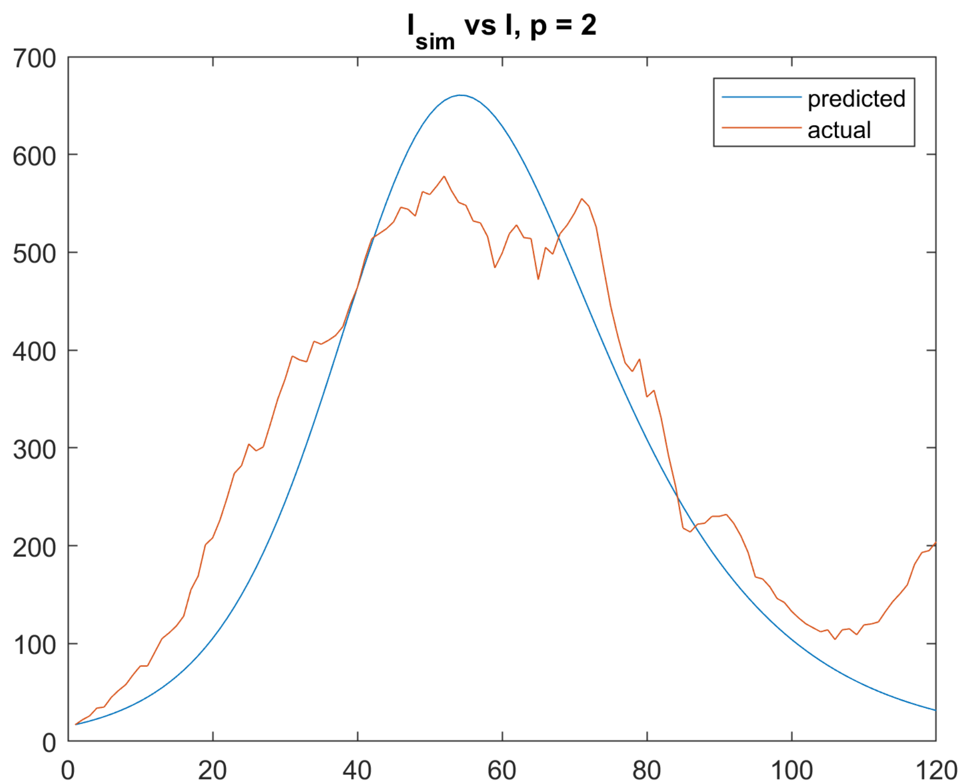
```
for pInd = 1:pLen
```

```
    p = pSet(pInd);
    % Get minimum parameter values
    paramMin = paramMinp(pInd,:);
    [alphaMin, betaMin, gammaMin, NMin] = paramMin{:};
    % Recalculated because it's not stored previously
    [Ssim, Isim, Rsim] = SIR_euler(I0, Tmax, alphaMin, betaMin, NMin);

    % (i) Compare Isim and I
    figure
    plot(Isim)
    hold on
    plot(I)
    hold off
    legend(["predicted", "actual"])
    title(['I_{sim} vs I, p = ', num2str(p)])

    % (ii) Compare Y and Ysim = gamma * Rsim
    figure
    plot(gammaMin * Rsim)
    hold on
    plot(Y(t0:t0+Tmax))
    hold off
    legend(["predicted", "actual"])
    title(['Y_{sim} vs Y, p = ', num2str(p)])
end
```

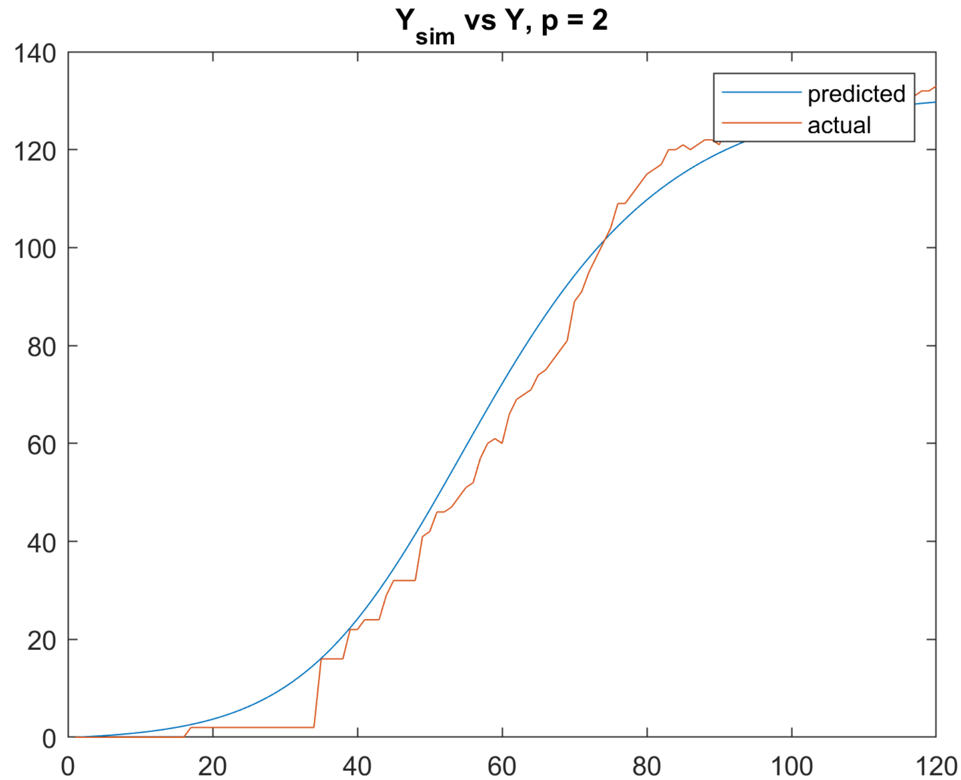


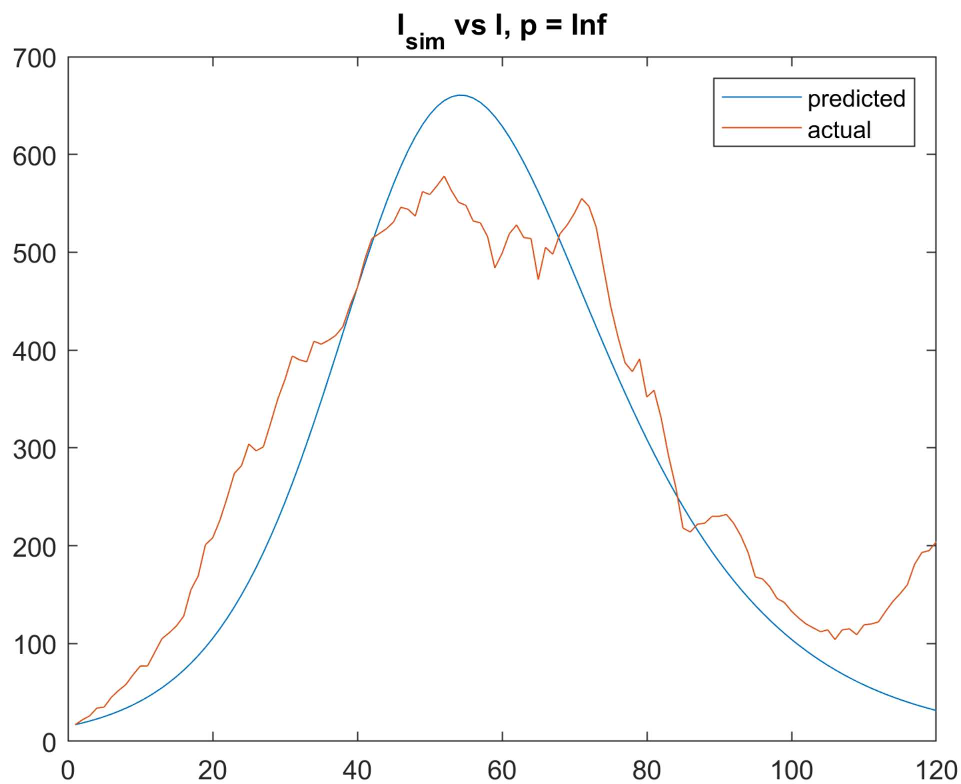


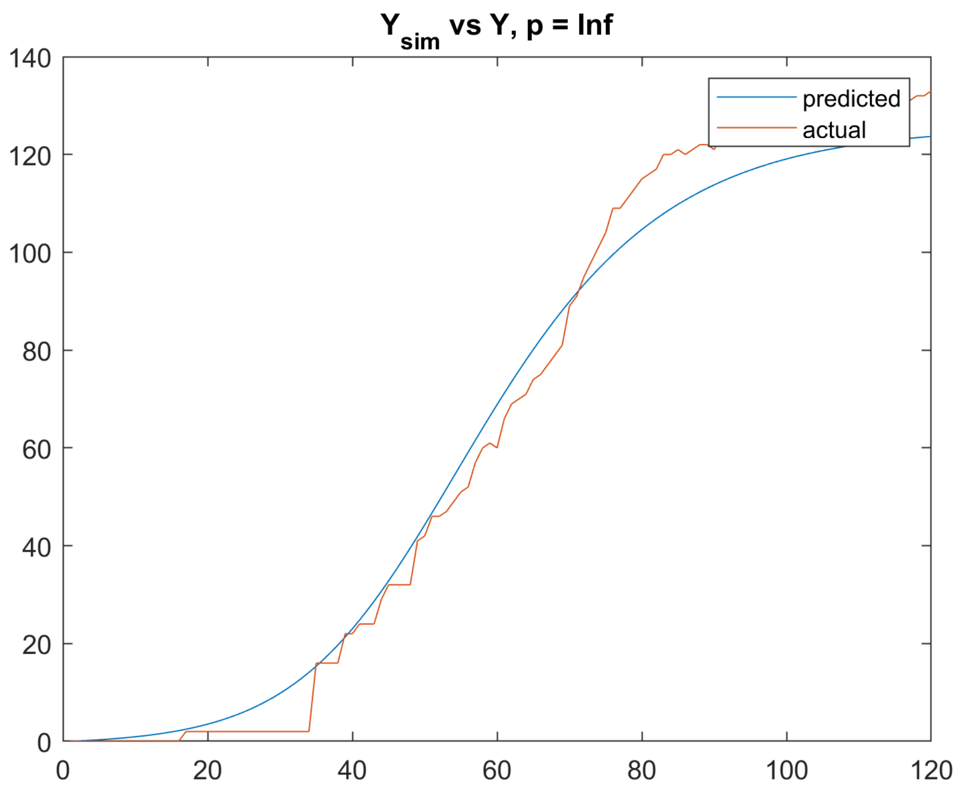
Y_sim vs Y, p = 1
predicted
actual
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120



I_sim vs I, p = 2
predicted
actual
700
600
500
400
300
200
100
0
0
20
40
60
80
100
120


Y_{sim} vs Y, p = 2
predicted
actual
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120


I_{sim} vs I, p = Inf
predicted
actual
700
600
500
400
300
200
100
0
0
20
40
60
80
100
120



Exercise One Part 4

Because for various *p* values, we got minima for alpha in {0.11}, for beta in {0.18, 0.19}, and for Nfrac in {0.02}, we would want to search in a range around those values to get more refined parameter estimates.

Surprisingly, all values of p returned Nfrac as 0.02 and alpha as 0.11, so perhaps this truly is a good value for N/alpha, or perhaps we need to search a smaller intervals around those values.

One possible set would be:

alpha in [0.10, 0.12]

R0 in [0.17, 0.20]

Nfrac in [0.01, 0.03].

We also would consider much smaller increments for the parameters instead of 0.01 / 0.1.

Exercise 2

```
% Get parameters
% Function defined in generateParams2.m
[alphaLen, betaLen, deltaLen, NLen, omega2] = ...
    generateParams2(alphaSet, deltaSet, R0Set, NfracSet, Nmax);
```

```
% Testing
% Euler scheme function defined in SEIR_euler.m
```

```
params = num2cell(squeeze(omega2(1,1,1,1,:)));
[alpha, beta, delta, N] = params{:}
```

```
alpha = 0.0500
beta = 0.0750
delta = 0.0500
N = 4.7368e+03
```

```
[Ssim, Esim, Isim, Rsim] = SEIR_euler(I0,Tmax,alpha,beta,delta,N)
```

```
Ssim = 120×1
10^3 ×
    4.7368
    4.7356
    4.7343
    4.7330
    4.7318
    4.7305
    4.7292
    4.7279
    4.7266
    4.7253
      ⋮
Esim = 120×1
   17.0000
   17.4059
   17.7930
   18.1637
   18.5198
   18.8632
   19.1955
   19.5180
   19.8321
   20.1389
      ⋮
Isim = 120×1
   17.0000
   17.0100
   17.0388
   17.0847
   17.1460
   17.2214
   17.3097
   17.4096
   17.5201
   17.6404
      ⋮
Rsim = 120×1
         0
    0.8502
    1.7013
    2.5543
    3.4100
    4.2691
    5.1323
    6.0002
    6.8734
    7.7523
      ⋮
```

```
[gamma, minVal] = minimizeGamma(t0, Tmax, Y, Rsim, 1)
```

```
gamma = 0.8901
minVal = 1.5965e+03
```

Exercise 2 Part 1

```
% gammas(alphaInd, betaInd, deltaInd, NInd, p) contains
% gamma as calculated to minimize p-norm of residual
% of actual results as compared to euler model
% with given parameters (alpha, beta, delta, N)
% J(...) contains the minimized function value
gammas = zeros(alphaLen, betaLen, deltaLen, NLen, pLen);
J = zeros(alphaLen, betaLen, deltaLen, NLen, pLen);

% WARNING: Takes a while to calculate
fprintf("alphas iterated over:")
```

```
alphas iterated over:
```

```
for alphaInd = 1:alphaLen
    fprintf("%.2f, ", alphaSet(alphaInd))
    for betaInd = 1:betaLen
        for deltaInd = 1:deltaLen
            for NInd = 1:NLen
                % Get parameters from set, use Euler scheme
                params = num2cell(squeeze(omega2(alphaInd, betaInd, deltaInd, NInd, :)));
                [alpha, beta, delta, N] = params{:};
                [Ssim, Esim, Isim, Rsim] = SEIR_euler(I0, Tmax, alpha, beta, delta, N);

                % For each p, find gamma minimizing p-norm & store
                for pInd = 1:pLen
                    p = pSet(pInd);
                    % Function defined in minimizeGamma.m
                    [gamma, minVal] = minimizeGamma(t0, Tmax, Y, Rsim, p);
                    gammas(alphaInd, betaInd, deltaInd, NInd, pInd) = gamma;
                    J(alphaInd, betaInd, deltaInd, NInd, pInd) = objectiveFunction(Y(t0:t0+Tmax
                end
            end
        end
    end
end
```

```
0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.20,
```

```
% Save results (so if we need to restart we don't lose the calculations)
% Commented so that we don't overwrite the results later
%writematrix(J,"hw4_ex2_errors_all_p.txt")
%writematrix(gammas,"hw4_ex2_gammas_all_p.txt")
```

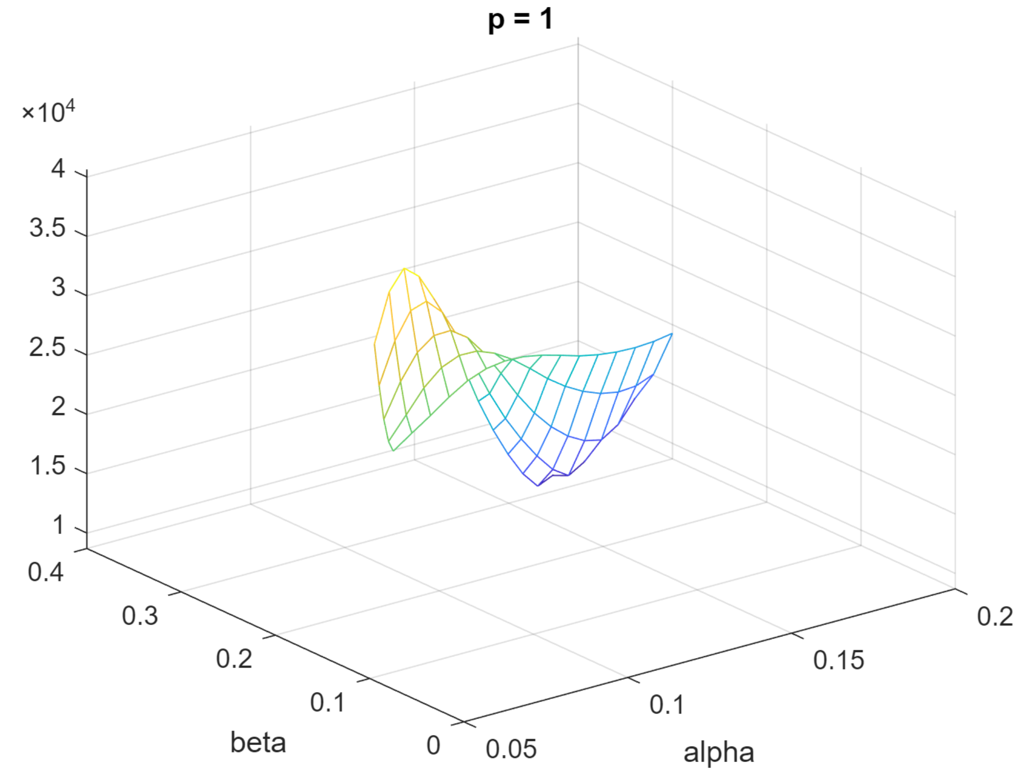
```
paramMinp = cell(pLen,5); % Store {alpha, beta, gamma, delta, N}
```

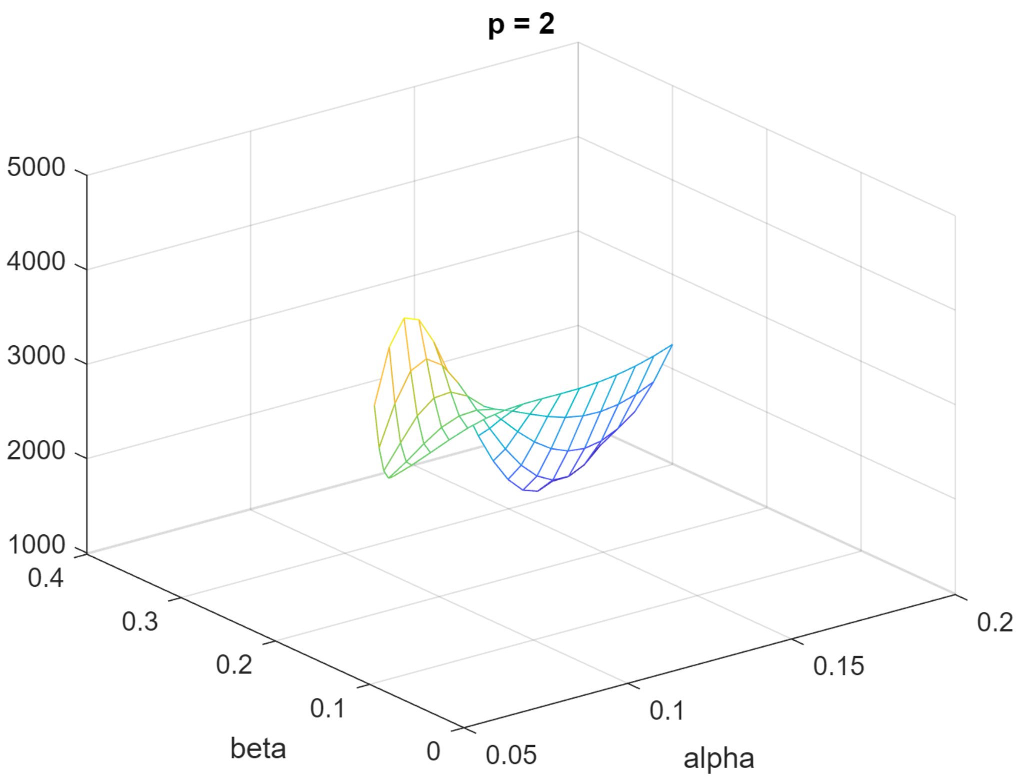
```
% For each p value...
for pInd = 1:pLen
    p = pSet(pInd);
    Jp = J(:,:,:,:,pInd);
    gammasp = gammas(:,:,:,:,pInd);

    % Get index of minimum error
    [M, Ind] = min(Jp, [], "all");
    [alphaMinInd, betaMinInd, deltaMinInd, NMinInd] = ind2sub(size(Jp),Ind);

    % Find parameters at that index
    gammaMin = gammasp(Ind);
    omegaTemp = reshape(omega2, [], 4);
    paramMin = num2cell(omegaTemp(Ind,:));
    [alphaMin, betaMin, deltaMin, NMin] = paramMin{:};

    % Print result
    fprintf("For p = %d, the parameters which reduce the error are\n" + ...
        "alpha = %.3f, beta = %.3f, gamma = %.3f, delta = %.3f, N = %g\n" + ...
        "(R0 = %.3f, Nfrac = %.3f)\n" + ...
        "with an error of %f\n\n", ...
        p,alphaMin, betaMin, gammaMin, deltaMin, NMin, betaMin / alphaMin, NMin / Nmax, M);
    paramMinp(pInd,:) = {alphaMin, betaMin, gammaMin, deltaMin, NMin};
end
```

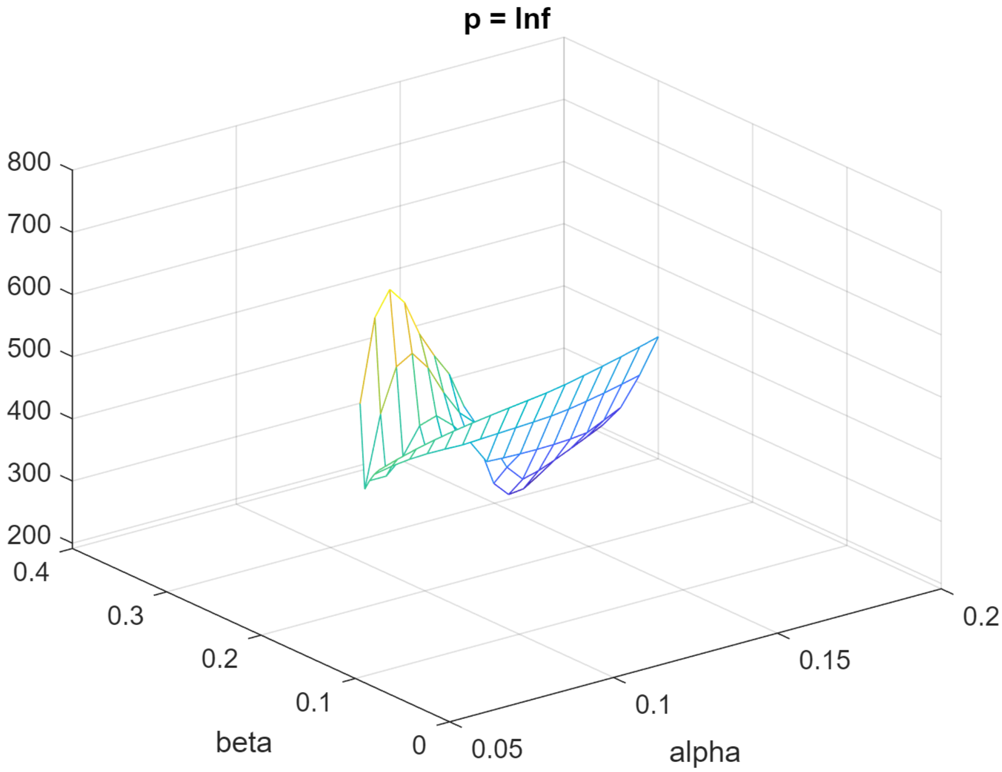
```
For p = 1, the parameters which reduce the error are
alpha = 0.170, beta = 0.306, gamma = 0.026, delta = 0.400, N = 7105.26
(R0 = 1.800, Nfrac = 0.030)
with an error of 8384.033963

For p = 2, the parameters which reduce the error are
alpha = 0.170, beta = 0.306, gamma = 0.026, delta = 0.400, N = 7105.26
(R0 = 1.800, Nfrac = 0.030)
with an error of 959.669060

For p = Inf, the parameters which reduce the error are
alpha = 0.170, beta = 0.323, gamma = 0.023, delta = 0.340, N = 7105.26
(R0 = 1.900, Nfrac = 0.030)
with an error of 191.415006
```

Exercise 2 Part 2

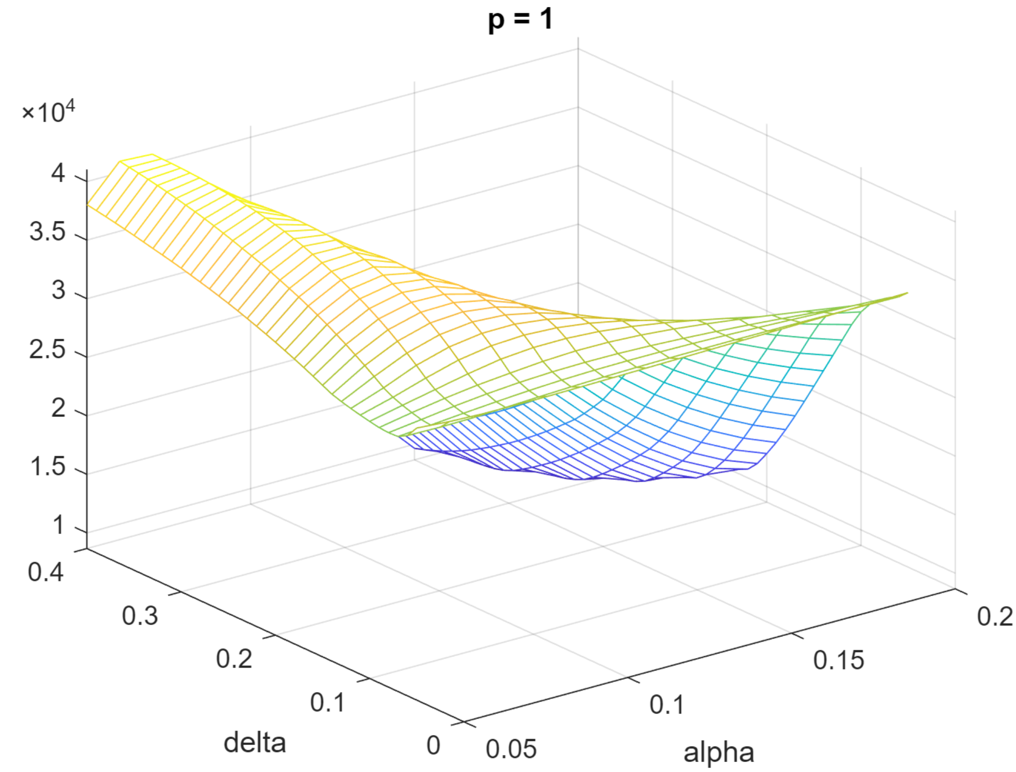
```
% (α, β) → J = J(α, β, δ-hat, N-hat, γ-hat)
for pInd = 1:pLen
    params = omega2(:, :, deltaMinInd, NMinInd, :);
    alphas = reshape(params(:,:,:,:,1), alphaLen, []);
    betas = reshape(params(:,:,:,:,2), [], betaLen);
    Jparams = J(:, :, deltaMinInd, NMinInd, pInd);

    figure
    mesh(alphas, betas, Jparams);
    title(['p = ', num2str(pSet(pInd))])
    xlabel("alpha"); ylabel("beta")
end
```

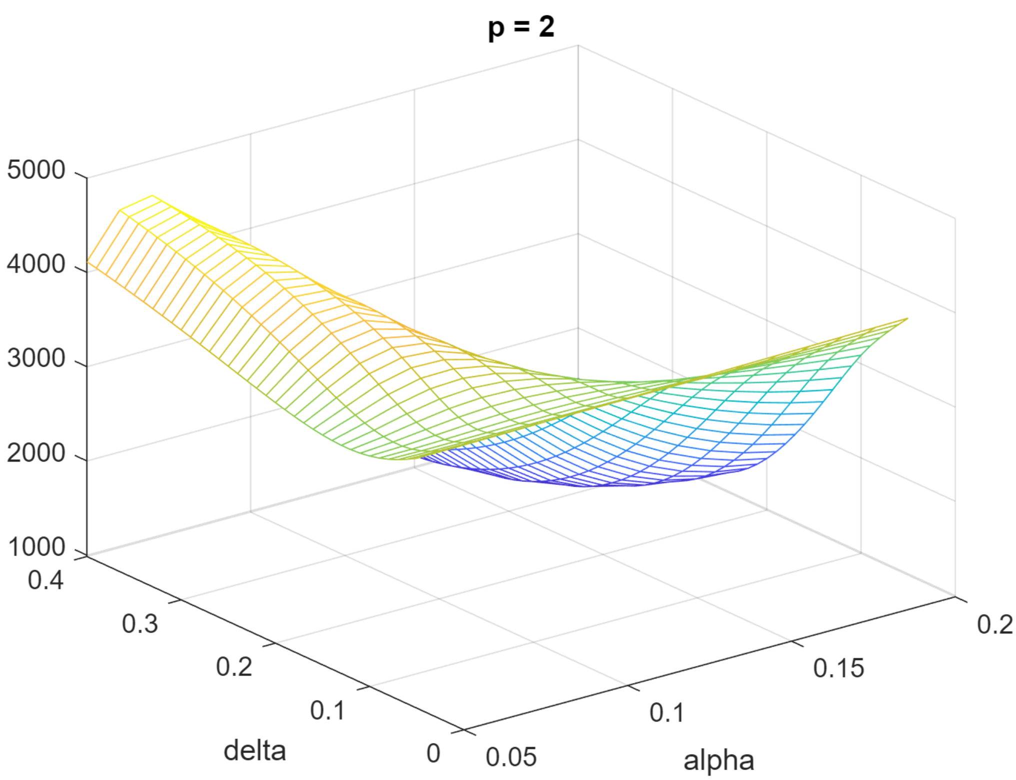

p = 1
×10^4
4
3.5
3
2.5
2
1.5
1
0.4
0.3
0.2
0.1
0
beta
0.05
0.1
0.15
0.2
alpha

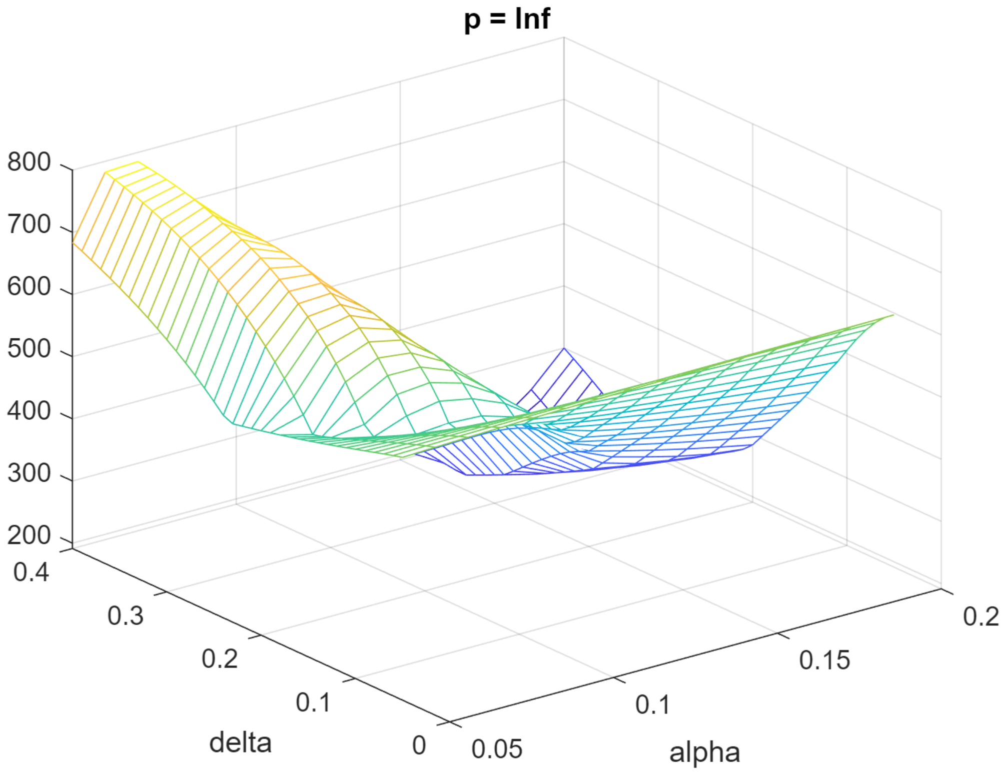

p = 2
5000
4000
3000
2000
1000
0.4
0.3
0.2
0.1
0
beta
0.05
0.1
0.15
0.2
alpha



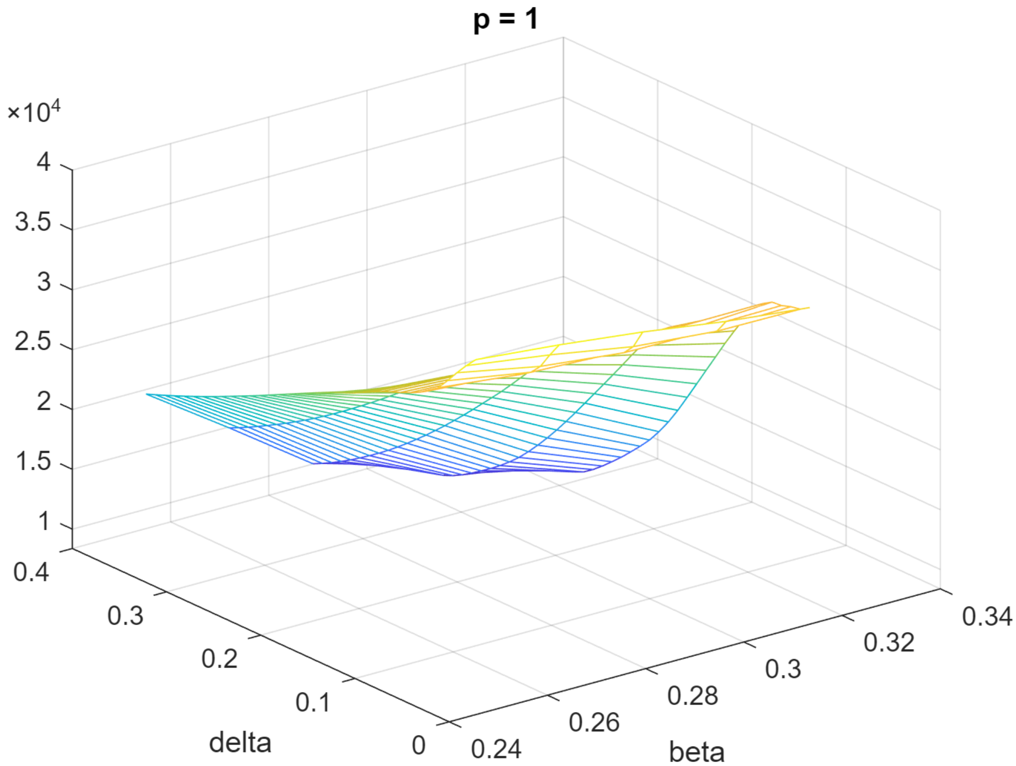

```
% (α, δ) → J = J(α, β-hat, δ, N-hat, γ-hat)
for pInd = 1:pLen
    params = omega2(:, betaMinInd, :, NMinInd, :);
    alphas = reshape(params(:,:,:,:,1), alphaLen, []);
    deltas = reshape(params(:,:,:,:,3), [], deltaLen);
    Jparams = reshape(J(:, betaMinInd, :, NMinInd, pInd), ...
        alphaLen, deltaLen);

    figure
    mesh(alphas, deltas, Jparams);
    title(['p = ', num2str(pSet(pInd))])
    xlabel("alpha"); ylabel("delta")
end
```

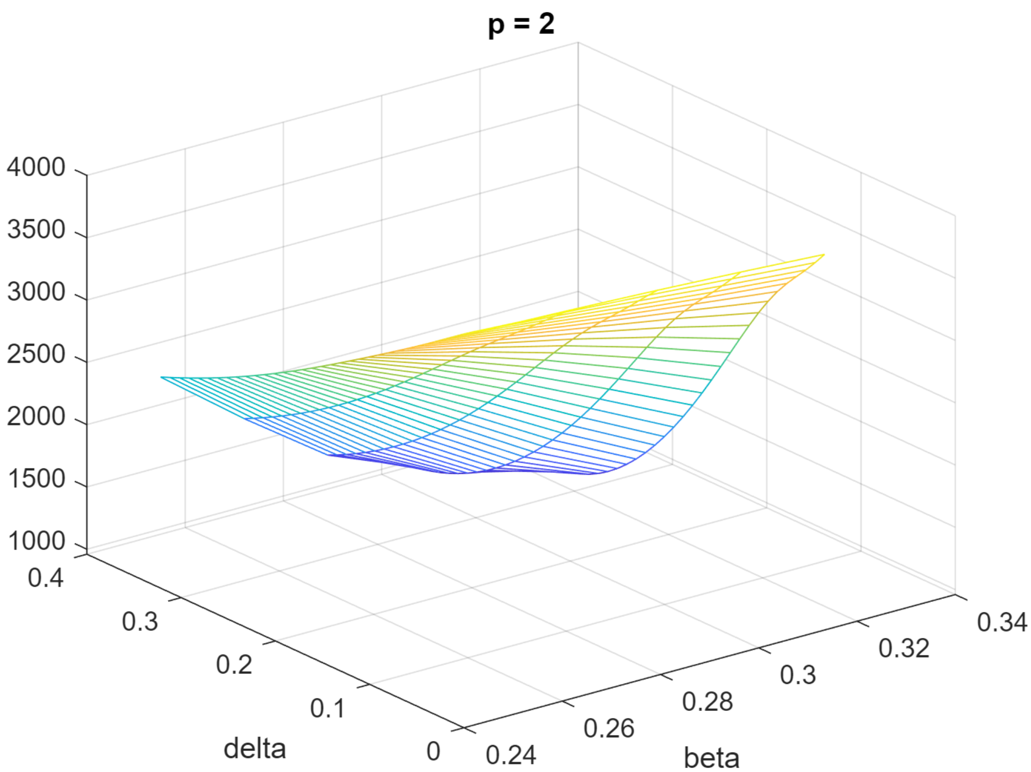

p = 1
×10^4
4
3.5
3
2.5
2
1.5
1
0.4
0.3
0.2
0.1
0
delta
0.05
0.1
0.15
0.2
alpha

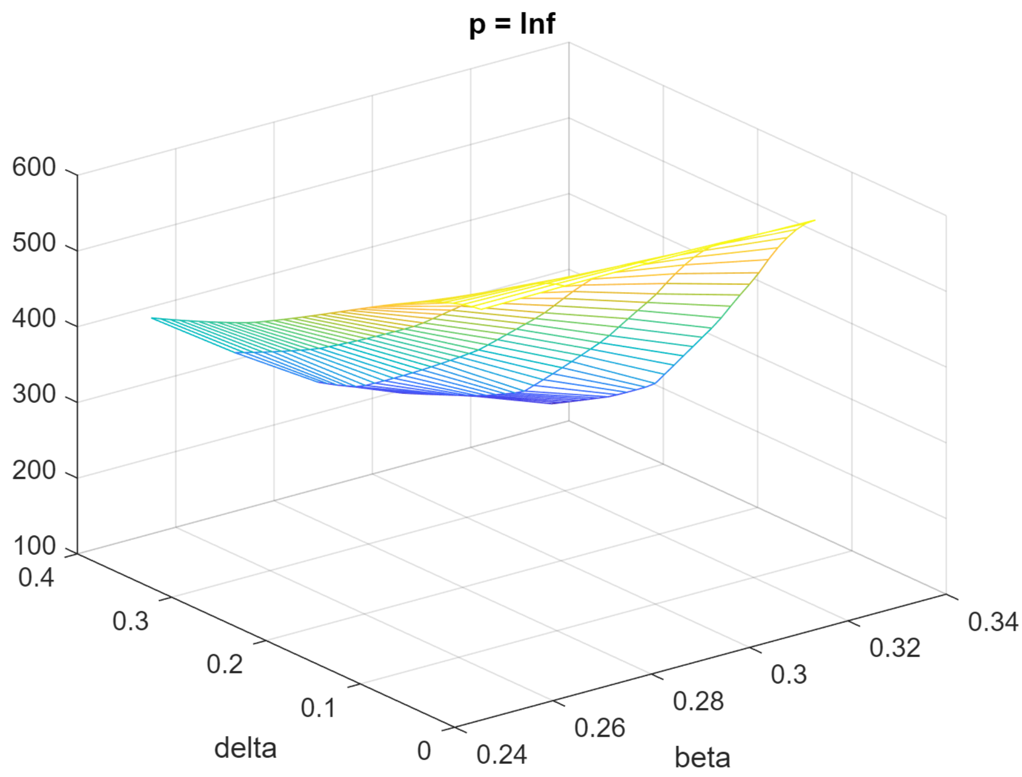

p = 2
5000
4000
3000
2000
1000
0.4
0.3
0.2
0.1
0
delta
0.05
0.1
0.15
0.2
alpha




```
% (β, δ) → J = J(α-hat, β, δ, N-hat, γ-hat)
for pInd = 1:pLen
    params = omega2(alphaMinInd, :, :, NMinInd, :);
    betas = reshape(params(:,:,:,:,2), betaLen, []);
    deltas = reshape(params(:,:,:,:,3), [], deltaLen);
    Jparams = reshape(J(alphaMinInd, :, :, NMinInd, pInd), ...
        betaLen, deltaLen);

    figure
    mesh(betas, deltas, Jparams);
    title(['p = ', num2str(pSet(pInd))])

    xlabel("beta"); ylabel("delta")
end
```


p = 1
×10^4
4
3.5
3
2.5
2
1.5
1
0.4
0.3
0.2
0.1
delta
0
0.24
0.26
0.28
0.3
0.32
0.34
beta

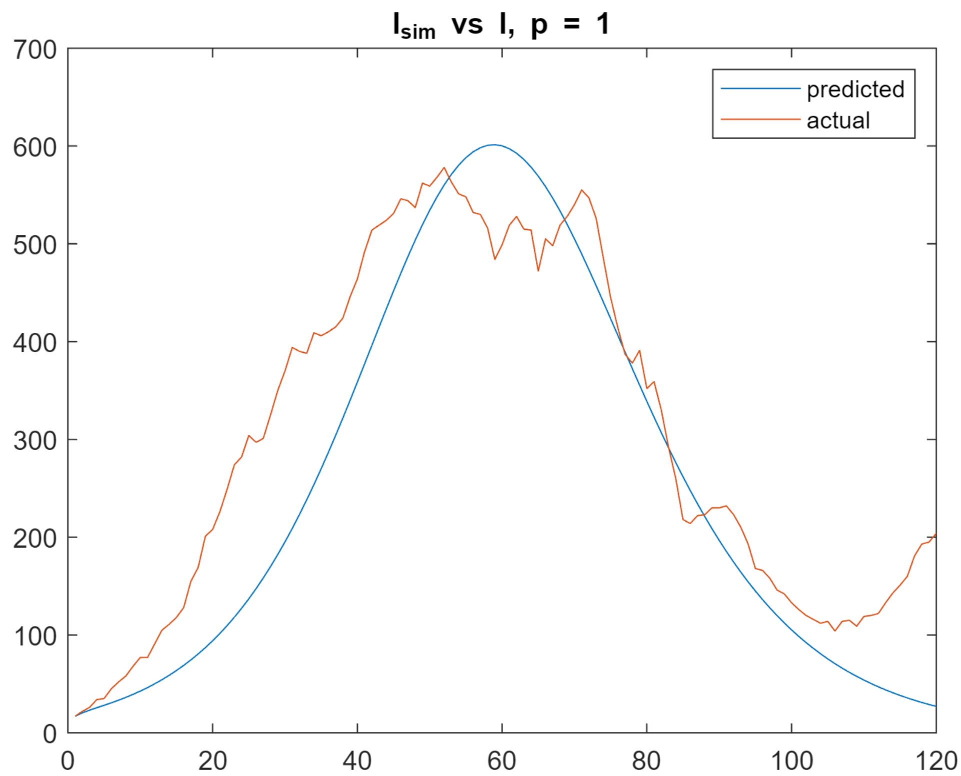

p = 2
4000
3500
3000
2500
2000
1500
1000
0.4
0.3
0.2
0.1
delta
0
0.24
0.26
0.28
0.3
0.32
0.34
beta

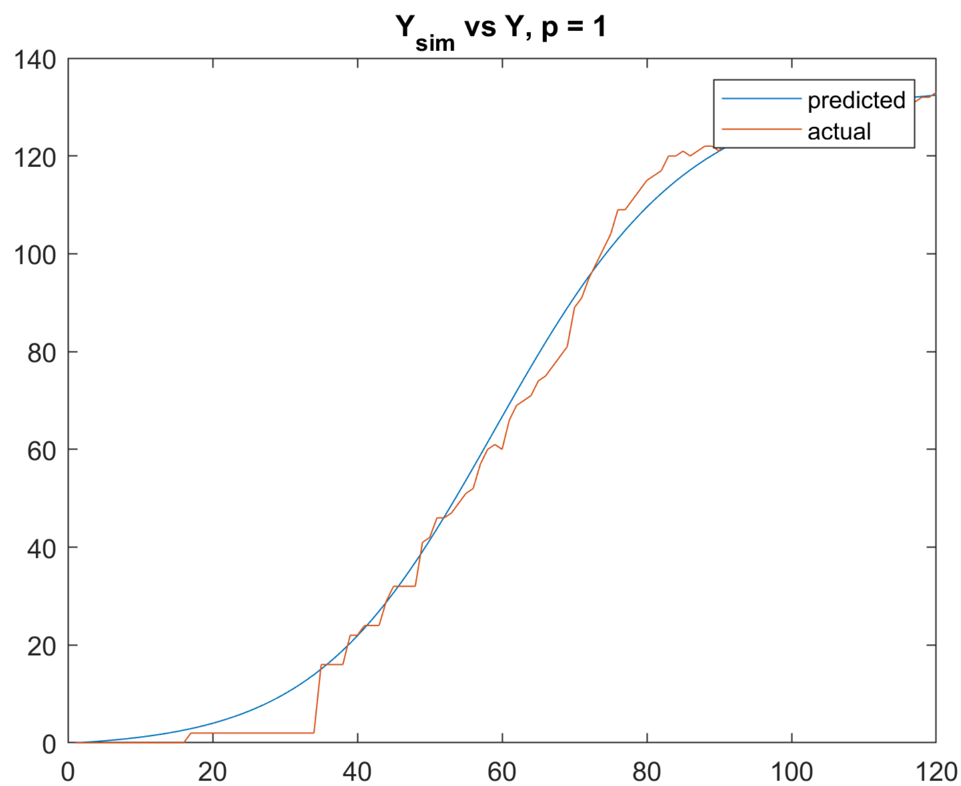


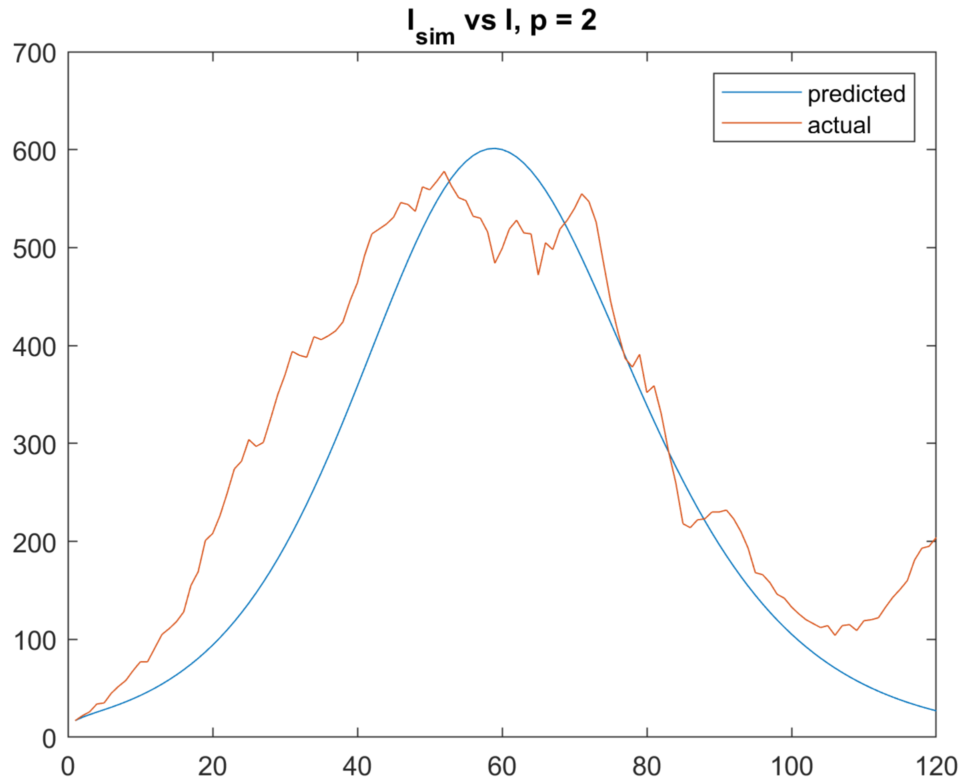
Exercise 2 Part 3

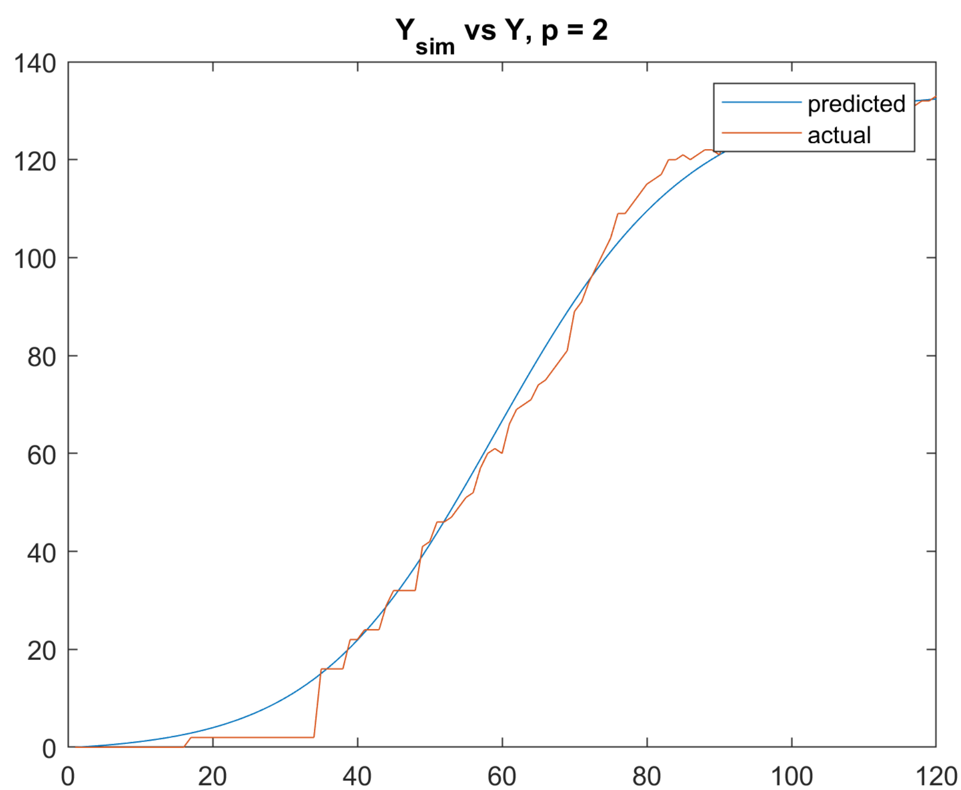
```
for pInd = 1:pLen
    p = pSet(pInd);
    % Get minimum parameter values
    paramMin = paramMinp(pInd,:);
    [alphaMin, betaMin, gammaMin, deltaMin, NMin] = paramMin{:};
    % Recalculated because it's not stored previously
    [Ssim, Esim, Isim, Rsim] = SEIR_euler(I0, Tmax, alphaMin, betaMin, deltaMin, NMin);

    % (i) Compare Isim and I
    figure
    plot(Isim)
    hold on
    plot(I)
    hold off
    legend(["predicted", "actual"])
    title(['I_{sim} vs I, p = ', num2str(p)])

    % (ii) Compare Y and Ysim = gamma * Rsim
    figure
    plot(gammaMin * Rsim)
    hold on
    plot(Y(t0:t0+Tmax))
    hold off
    legend(["predicted", "actual"])
    title(['Y_{sim} vs Y, p = ', num2str(p)])
```

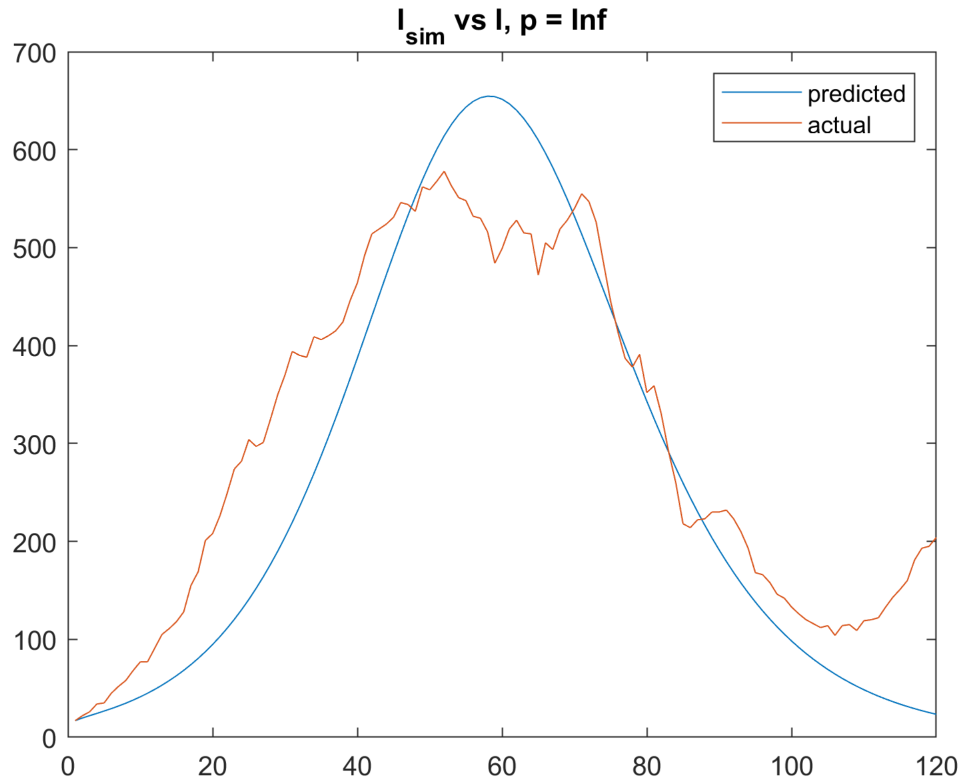
```
end
```

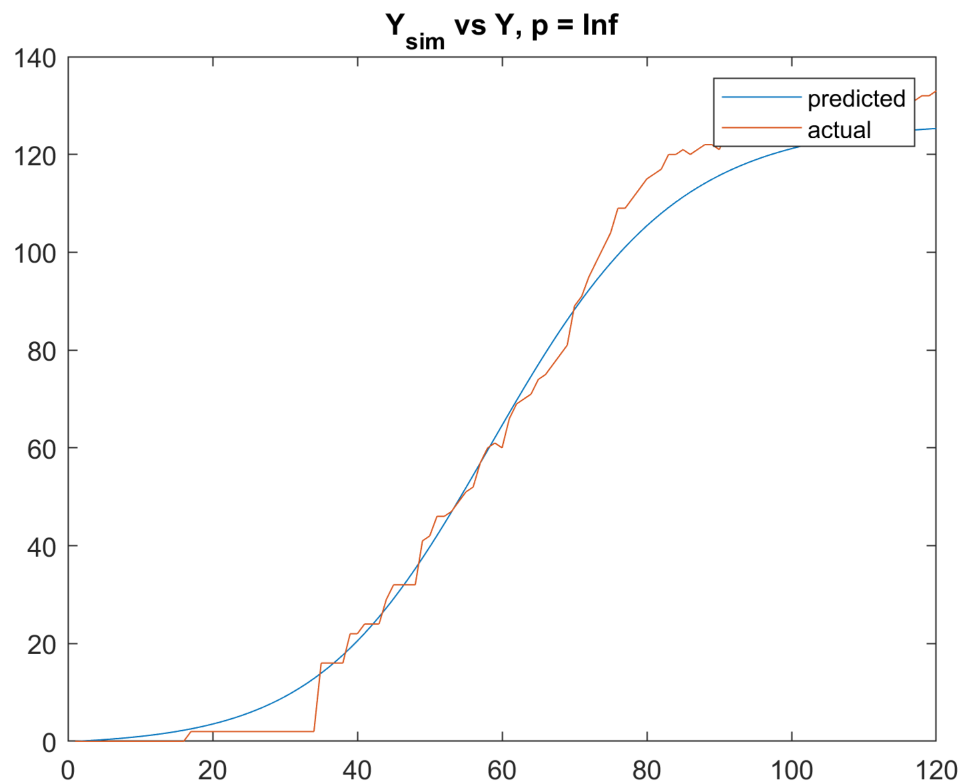





I_{sim} vs I, p = 2
700
600
500
400
300
200
100
0
0
20
40
60
80
100
120
predicted
actual


Y_{sim} vs Y, p = 2
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
predicted
actual





Exercise 2 Part 4

Because for various *p* values, we got minuma for alpha in {0.17}, for R0 in {1.8, 1.9}, for delta in {0.34, 0.40}, and for Nfrac in {0.03}, we would want to search in a range around those values to get more refined parameter estimates. One possible set would be:

alpha in [0.16, 0.18]

R0 in [1.7, 2.0]

delta in [0.33, 0.41]

Nfrac in [0.02, 0.04]

We also would consider much smaller increments for the parameters instead of 0.01 / 0.1.