```matlab
%% Q1 MLR
% Set-up
tutorial_setup

% Populate A and b matrix
b = T_data';
A(:, 1) = 1;
tri = 0:dt_data/2:1;
tri = flip(tri)';
[trilen, ~] = size(tri);

% Triangle of width 2s
for c = 2:1:4
    for ii = 1:1:rows
        if t_impulse(c-1) <= t_meas(ii)
            A(ii:ii+trilen-1, c) = tri;
            break
        end
    end
end

theta = A\b;
k = theta(1); J = theta(2:end) * V * Cp;
fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J(1), J(2), J(3));

% Populate A and b matrix
b = T_data';
A = zeros(rows, cols);
A(:, 1) = 1;

% Exponentially decaying: exp(-2*(t-t_impulse(x))
for c = 2:1:4
  for ii = 1:1:rows
    if t_impulse(c-1) <= t_meas(ii)
      A(ii:end, c) = exp(-2*(t_meas(ii:end) - t_meas(ii)));
      break
    end
  end
end

theta = A\b;
k = theta(1); J = theta(2:end) * V * Cp;
fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J(1), J(2), J(3));

%% Q2 IIM

% Set-up
tutorial_setup

n_loops = 100;
dt_ratio = 10;
dt_HR = dt_data/dt_ratio;  % High resolution time delta
t_HR = t_start:dt_HR:t_end;  % High resolution time array
[~, N_HR] = size(t_HR);
```

```

% Matrix initialization
phi = zeros(N_HR, cols-1);
psi = zeros(1, n_loops);

% Populate A and b matrix
T0 = mean(T_data(1:2));

for c = 1:1:3
  for ii = 1:1:N_HR
    if t_impulse(c) <= t_HR(ii)
      phi(ii, c) = 1/dt_HR;
      A(ceil(ii/dt_ratio)+1:end, c+1) = 1;
      break
    end
  end
end


% Simulate and iterate
set(0,'defaultfigureposition',[60 60 420,330])
figure(1)
set(gcf,'position',[60 60 680 330])
h1=axes('position',[.085 .1 .40 .85]);hold all
h2=axes('position',[.585 .1 .40 .85]);hold all

figure(2)
h3=axes;hold all
yyaxis left; hold all;
plot(h3, [0 n_loops], [parent(2) parent(2)]/1000, 'b-');
plot(h3, [0 n_loops], [parent(3) parent(3)]/1000, 'g-');
plot(h3, [0 n_loops], [parent(4) parent(4)]/1000, 'r-');
yyaxis right; hold all;
plot(h3, [0 n_loops], [parent(1) parent(1)], 'k-');
plot(h1, t_meas, T_data, '+k');

% Populate b matrix and make initial guess
T0 = mean(T_data(1:2));
T_fwd = zeros(size(t_HR)); % t_HR = 0:0.025:10 [1x401]
b = T_data' - T0;

for ii = 1:n_loops  % 1:100
  % Update A column 1 (only one that changes), calculate theta
  iTg = cumtrapz(t_HR, -(T_fwd - T_amb));
  A(:, 1) = iTg(1:dt_ratio:end)';  % dt_ratio = 10 -> iTg(1:10:401)' [41x1]
  theta = A\b;

  % Calculate parameter values and print
  k = theta(1);
  J1 = theta(2)*V*Cp; J2 = theta(3)*V*Cp; J3 = theta(4)*V*Cp;
  fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J1, J2, J3);

  % Forward simulate and store residuals
  % Analytical
```

```
    T_fwd = exp(-k*t_HR).*(T0+cumtrapz(t_HR, exp(k*t_HR).* ...
      ((phi * theta(2:end)) + (k*T_amb))'));

    % Picard: Error stepping, doesnt work
%   T_fwd = T0 + cumtrapz(t_HR, -k*(T_fwd-T_amb) + (phi*theta(2:end))');
%   % Works but doesnt make sense
%   for jj = 2:N_HR  % 2:401
%     T_fwd(1:jj) = T0 + cumtrapz(t_HR(1:jj), -k*(T_fwd(1:jj)-T_amb) + (phi(1:jj, :)*theta(2:end))');
%   end

    % Time-stepping: Euler 1st order
%   T_fwd(1) = T0;
%   for jj = 2:N_HR  % 2:401
%     T_fwd(jj) = T_fwd(jj-1) + (-k*(T_fwd(jj-1)-T_amb) + (phi(jj-1,:)*theta(2:end))')*dt_HR;
%   end

    % Store residuals
    psi(ii) = norm(T_fwd(1:dt_ratio:end)-T_data);

    % Plot forward simulation every 5th iteration
    if mod(ii, 5) == 0
      plot(h1, t_HR, T_fwd, 'r:');ylim(h1, [20, 45]);
      ylabel(h1, 'Temp (^oC)'); xlabel(h1, 'Time (s)');
    end

    % Plot Residuals
    plot(h2, ii, psi(ii), 'ob','markerfacecolor','b');
    ylabel(h2, '\bfresiduals \it\psi'); xlabel(h2, 'Iterations');

    % Plot paramater convergence
    yyaxis left;  % J1, J2, J3 on left axis
    plot(h3, ii, J1/1000, '.b', ii, J2/1000, '.g', ii, J3/1000, '.r');
    ylabel(h3, 'J_x (kJ)'); xlabel(h3, 'Iterations'); ylim(h3, [0, 600]);

    yyaxis right; % k on right axis
    plot(h3, ii, k, '.k');
    ylabel(h3, 'k (s^-^1)'); xlabel(h3, 'Iterations'); ylim(h3, [0, 5]);
end

plot(h1, t_HR, T_fwd, 'b.');ylim(h1, [20, 45]);
ylabel(h1, 'Temp (^oC)'); xlabel(h1, 'Time (s)');


%% Q3 ARX

tutorial_setup

% Part a

[~, n_data] = size(T_data);

b = T_data(2:end)';
```

```
A = zeros(n_data-1, 5);
A(:, 1) = T_data(1:end-1)';
A(:, 5) = 1;

for c = 2:1:4
  for ii = 1:1:n_data
    if t_impulse(c-1) <= t_meas(ii)
      A(ii-1, c) = 1;
      break
    end
  end
end

theta = A\b;
spy(A)

T_arx1 = zeros(size(T_data));
T_arx1(1) = T_data(1);
for t = 2:n_data
  T_arx1(t) = theta(1) * T_arx1(t-1) + theta(2:4)' * A(t-1, 2:4)' + theta(5);
end

figure(302);
plot(t_meas, T_arx1, 'r', t_meas, T_data, '+k');
ylim([20, 45]); xlabel('Time [s]'); ylabel('Temperature [^oC]');
hold on;

% Part b
t_s = 0:0.25:3;
[~, rows] = size(t_s);
phi = A(:, 2) + 2*A(:, 3) + 1.5*A(:, 4);  % Combine inputs into one fn
A2 = zeros(rows, 3);
A2(:, 1) = T_data(1:rows);
A2(:, 2) = A(1:rows, 2); A2(end, 2) = 2;
A2(:, 3) = 1;
b2 = T_data(2:rows+1)';

theta2 = A2\b2;  % Used for simulating, not param ID

T_arx2 = zeros(size(T_data));
T_arx2(1) = T_data(1);

for t = 2:n_data
  T_arx2(t) = theta2(1)*T_arx2(t-1) + theta2(2)*phi(t-1) + theta2(3);
end

plot(t_meas, T_arx2, 'b');

%% Q4 Gradient Descent

tutorial_setup

% Forward simulation function
Temp_f = @(ts, t0, th, ph) exp(-th(1)*ts) .* (t0+cumtrapz(ts, ...
```

```
  exp(th(1)*ts).*((ph*th(2:end))+(th(1)*T_amb))'));

% Initializations
[~, n_data] = size(T_data);
J = zeros(n_data, cols);
n_iterations = 30;
T0 = mean(T_data(1:2));

dt_HR = 0.025;
t_HR = 0:dt_HR:10;
n_HR = 10/dt_HR;
phi = zeros(size(t_HR'));

for c = 1:1:3
  for ii = 1:1:n_HR
    if t_impulse(c) <= t_HR(ii)
      phi(ii, c) = 1/dt_HR;
      break
    end
  end
end

theta = zeros(cols, n_iterations);
psi = zeros(rows, 1);
TT = 0:dt_data/dt_HR:n_HR;

% Initial guess
delta = [0.05, 100, 10, 10]';  % Pertubations
lambda = 1;

% Guess theta(1), calculate psi(1)
theta(:, 1) = [0 0 0 0]';  % Initial conditions/guess
T_fwd = Temp_f(t_HR, T0, theta(:, 1), phi);
psi(:, 1) = T_fwd(TT+1)' - T_data';

for it = 1:n_iterations-1
  % Calculate J: same for all Gauss-Newton, Levenberg, Levenberg-Marquardt
  for col = 1:4
    dtheta = zeros(4,1); dtheta(col, 1) = delta(col, 1);
    dpsidth_i =  (Temp_f(t_HR, T0, theta(:, it) + dtheta, phi) - T_fwd)/delta(col);
    J(:, col) = dpsidth_i(TT+1)';
  end
  % Next part is different for next iteration of theta!
  % Levenberg
  % theta(:, it+1) = theta(:, it) - (((J'*J)+lambda*eye(4))^-1) * (J'*psi(:, it));
  % T_fwd = Temp_f(t_HR, T0, theta(:, it+1), phi);
  % psi(:, it+1) = T_fwd(TT+1)' - T_data';

  % Gauss-Newton
  % theta(:, it+1) = theta(:, it) - ((J'*J)^-1)*(J'*psi(:, it));
  % T_fwd = Temp_f(t_HR, T0, theta(:, it+1), phi);
  % psi(:, it+1) = T_fwd(TT+1)' - T_data';
```

```

  % Levenberg-Marquardt
  while true
    % Calculate theta(ii+1)
    theta(:, it+1) = theta(:, it) - (((J'*J)+lambda*eye(4).*(J'*J))^-1) *(J'*psi(:, it));

    % Forward simulate for psi(ii+1)
    T_fwd = Temp_f(t_HR, T0, theta(:, it+1), phi);
    psi(:, it+1) = T_fwd(TT+1)' - T_data';

    % Levenberg-Marquardt
    % Update lambda, repeat if norm(psi(ii+1)) > norm(psi(ii))
    if norm(psi(:, it+1)) <= norm(psi(:, it))
      lambda = lambda / 5;
      break
    else
      lambda = lambda * 5;
    end
  end

  % Calculate the Paramaters to ID
  k = theta(1, it+1);
  Jx = theta(2:end, it+1)*V*Cp;
  fprintf('GN: k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, Jx(1), Jx(2), Jx(3));
end

fprintf('\n');

% Steepest Descent
n_iterationsSD = 20000;  % Takes a real fucking long time
thetaSD = zeros(cols, n_iterationsSD); thetaSD(:, 1) = [10 1 1 1]';
deltaSD = theta(:, end)/1000;
JSD = zeros(cols, 1);
alpha = 0.01;

for it = 1:n_iterationsSD-1
  T_fwdSD = Temp_f(t_HR, T0, thetaSD(:, it), phi);
  psi_thi = norm(T_fwdSD(TT+1)' - T_data');

  for ii = 1:4
    dtheta = zeros(4,1); dtheta(ii, 1) = deltaSD(ii, 1);
    T_fwdSD_i = Temp_f(t_HR, T0, thetaSD(:, it) + dtheta, phi);
    [T_fwdSD(TT+1)' T_fwdSD_i(TT+1)' T_data'];
    psi_thi_i =  norm(T_fwdSD_i(TT+1)' - T_data');
    dpsi_dth_i = (psi_thi_i - psi_thi)/deltaSD(ii);
    [dpsi_dth_i psi_thi_i psi_thi];
    JSD(ii, 1) = dpsi_dth_i;  % [4x1]
  end
  JSD;

  thetaSD(:, it+1) = thetaSD(:, it) - alpha*JSD;
  k = thetaSD(1, it+1);
  Jx = thetaSD(2:end, it+1)*V*Cp;
```

```
  fprintf('SD: k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, Jx(1), Jx(2), Jx(3));
end

figure(403);
yyaxis left;
plot([0 n_iterationsSD], [parent(2) parent(2)]/1000, 'k', ...
    [0 n_iterationsSD], [parent(3) parent(3)]/1000, 'k', ...
    [0 n_iterationsSD], [parent(4) parent(4)]/1000, 'k');
hold on
plot(1:n_iterationsSD, thetaSD(2, :)*V*Cp/1000, '.b', ...
  1:n_iterationsSD, thetaSD(3, :)*V*Cp/1000, '.b', ...
  1:n_iterationsSD, thetaSD(4, :)*V*Cp/1000, '.b');
ylim([0, 550]);
xlabel('Iterations');ylabel('J_x [kJ]')
hold all;

yyaxis right;
plot(1:n_iterationsSD, thetaSD(1, :), '.r');
plot([0 n_iterationsSD], [parent(1) parent(1)], 'r-');
xlabel('Iterations');ylabel('k [s^-^1]')
ylim([0, 4]);

figure(401);
yyaxis left;
plot([0 n_iterations], [parent(2) parent(2)]/1000, 'k', ...
    [0 n_iterations], [parent(3) parent(3)]/1000, 'k', ...
    [0 n_iterations], [parent(4) parent(4)]/1000, 'k');
hold on
plot(1:n_iterations, theta(2, :)*V*Cp/1000, '.-b', ...
  1:n_iterations, theta(3, :)*V*Cp/1000, '.b', ...
  1:n_iterations, theta(4, :)*V*Cp/1000, ':b');
ylim([0, 550]);
xlabel('Iterations');ylabel('J_x [kJ]')
hold all;

yyaxis right;
plot(1:n_iterations, theta(1, :), '.r');
plot([0 n_iterations], [parent(1) parent(1)], 'r-');
xlabel('Iterations');ylabel('k [s^-^1]')
ylim([0, 4]);

figure(402);
plot(t_HR, T_fwd, 'ob', t_HR, T_fwdSD, '--r'); ylim([20, 45]);
ylabel('Temp (^oC)'); xlabel('Time (s)');


%% Q5 Genetic Algorithm

tutorial_setup

% Initialize forward simulation variables
T0 = mean(T_data(1:2));
dt_ratio = 25;
dt_HR = dt_data/dt_ratio;
```

```
t_HR = t_start:dt_HR:t_end;
n_HR = t_end/dt_HR;
phi = zeros(size(t_HR'));

for c = 1:1:3
  for ii = 1:1:n_HR
    if t_impulse(c) <= t_HR(ii)
      phi(ii, c) = 1/dt_HR;
      break
    end
  end
end

% Forward simulation function
Temp_f = @(ts, t0, th, ph) exp(-th(1)*ts) .* (t0+cumtrapz(ts, ...
  exp(th(1)*ts).*((ph*th(2:end))+(th(1)*T_amb))'));

% Initialize
generations = 250;
K = 100;
N = 10;
mut_rate = 0.05;
thetas = 30*rand(K, 5);
max_err = zeros(1, generations);

% Create plots
figure(501);

% K plot
ax_k = subplot(231); hold all;
xlabel(ax_k, 'Generations'); ylabel(ax_k, '\Theta_{opt}(k) [s^{-1}]');
plot(ax_k, [0 generations], [parent(1) parent(1)], '-g');
xlim(ax_k, [0, generations]); ylim(ax_k, [0 6]);

% J1, J2, J3 plot
ax_j = subplot(232); hold all;
xlabel(ax_j, 'Generations'); ylabel(ax_j, '\Theta_{opt}(J_x) [kJ]');
plot(ax_j, [0 generations], [parent(2) parent(2)]/1000, '-.r');
plot(ax_j, [0 generations], [parent(3) parent(3)]/1000, '-.b');
plot(ax_j, [0 generations], [parent(4) parent(4)]/1000, '-.m');
xlim(ax_j, [0, generations]); ylim(ax_j, [0 600]);

% Residuals plot
ax_ps = subplot(233); hold all;
xlabel(ax_ps, 'Generations'); ylabel(ax_ps, '\Psi_{opt}');
xlim(ax_k, [0, generations]);

% Forward simulation plot with theta opt
ax_sim = subplot(212); hold all;
xlabel(ax_sim, 'Time [s]'); ylabel(ax_sim, 'Temperature [^oC]');
ylim(ax_sim, [20 45]);

% Loop for each generation
for gen = 1:generations
```

```

% Iterate through parameter sets
for ii = 1:K
  % Determine psi of theta(:, i)
  theta = thetas(ii, 2:end)';
  T_fwd = Temp_f(t_HR, T0, theta, phi);
  thetas(ii, 1) = norm(T_fwd(1:dt_ratio:end)' - T_data');
end

% Sort thetas by the lowest (best) residuals
thetas = sortrows(thetas);

% Check for NaNs
for ii = 1:K
  if isnan(thetas(ii, 1)) || isnan(thetas(ii, 1))
    % Create an iterator of length K for only "good" organisms
    ind=1; jj=1; iterator = zeros(1, K);
    while jj <= 100
      iterator(jj) = ind;
      jj = jj + 1;
      if mod(jj, ceil(K/ii)) == 0
       ind=ind+1;
      end
    end

    % Appropriately remove NaN and Inf sets, replace with better clones
    thetas(:, :) = thetas(iterator, :);
    break
  end
end

% Discard the worst and clone multiple copies of the top ranking sets
thetas(:, :) = thetas(ceil(((1:K)/20).^2), :);

% Mutate, keep one copy of
thetas(2:end, 2:end) = thetas(2:end, 2:end) + ...
  mut_rate*[thetas(2:end,1) thetas(2:end,1) thetas(2:end,1) thetas(2:end,1)].*randn(K-1,4);

% Plot of theta opt
if mod(gen, 10) == 1
  T_fwd = Temp_f(t_HR, T0, thetas(1, 2:end)', phi);
  plot(ax_sim, t_HR, T_fwd, 'r');
end

% Plot optimal values from this generation
ps = thetas (1,1);

k = thetas(1,2); J = thetas(1,3:end)*V*Cp;
fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J(1), J(2), J(3));

plot(ax_k, gen, k, '.g');
plot(ax_j, gen, J(1)/1000, '.r', gen, J(2)/1000, '.b', gen, J(3)/1000, '.m');
plot(ax_ps, gen, ps, '.k');
```

```
end


plot(ax_sim, t_HR, T_fwd, 'k');
plot(ax_sim, t_meas, T_data, '+b');

```