

```
1 %% Q1 MLR
2 % Set-up
3 tutorial_setup
4
5 % Populate A and b matrix
6 b = T_data';
7 A(:, 1) = 1;
8 tri = 0:dt_data/2:1;
9 tri = flip(tri)';
10 [trilen, ~] = size(tri);
11
12 % Triangle of width 2s
13 for c = 2:1:4
14     for ii = 1:1:rows
15         if t_impulse(c-1) <= t_meas(ii)
16             A(ii:ii+trilen-1, c) = tri;
17             break
18         end
19     end
20 end
21
22 theta = A\b;
23 k = theta(1); J = theta(2:end) * V * Cp;
24 fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J(1), J(2), J(3));
25
26 % Populate A and b matrix
27 b = T_data';
28 A = zeros(rows, cols);
29 A(:, 1) = 1;
30
31 % Exponentially decaying: exp(-2*(t-t_impulse(x))
32 for c = 2:1:4
33     for ii = 1:1:rows
34         if t_impulse(c-1) <= t_meas(ii)
35             A(ii:end, c) = exp(-2*(t_meas(ii:end) - t_meas(ii)));
36             break
37         end
38     end
39 end
40
41 theta = A\b;
42 k = theta(1); J = theta(2:end) * V * Cp;
43 fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J(1), J(2), J(3));
44
45 %% Q2 IIM
46
47 % Set-up
48 tutorial_setup
49
50 n_loops = 100;
51 dt_ratio = 10;
52 dt_HR = dt_data/dt_ratio; % High resolution time delta
53 t_HR = t_start:dt_HR:t_end; % High resolution time array
54 [~, N_HR] = size(t_HR);
```

```
55
56 % Matrix initialization
57 phi = zeros(N_HR, cols-1);
58 psi = zeros(1, n_loops);
59
60 % Populate A and b matrix
61 T0 = mean(T_data(1:2));
62
63 for c = 1:1:3
64     for ii = 1:1:N_HR
65         if t_impulse(c) <= t_HR(ii)
66             phi(ii, c) = 1/dt_HR;
67             A(ceil(ii/dt_ratio)+1:end, c+1) = 1;
68             break
69         end
70     end
71 end
72
73
74 % Simulate and iterate
75 set(0,'defaultfigureposition',[60 60 420,330])
76 figure(1)
77 set(gcf,'position',[60 60 680 330])
78 h1=axes('position',[.085 .1 .40 .85]);hold all
79 h2=axes('position',[.585 .1 .40 .85]);hold all
80
81 figure(2)
82 h3=axes;hold all
83 yyaxis left; hold all;
84 plot(h3, [0 n_loops], [parent(2) parent(2)]/1000, 'b-');
85 plot(h3, [0 n_loops], [parent(3) parent(3)]/1000, 'g-');
86 plot(h3, [0 n_loops], [parent(4) parent(4)]/1000, 'r-');
87 yyaxis right; hold all;
88 plot(h3, [0 n_loops], [parent(1) parent(1)], 'k-');
89 plot(h1, t_meas, T_data, '+k');
90
91 % Populate b matrix and make initial guess
92 T0 = mean(T_data(1:2));
93 T_fwd = zeros(size(t_HR)); % t_HR = 0:0.025:10 [1x401]
94 b = T_data' - T0;
95
96 for ii = 1:n_loops % 1:100
97     % Update A column 1 (only one that changes), calculate theta
98     iTg = cumtrapz(t_HR, -(T_fwd - T_amb));
99     A(:, 1) = iTg(1:dt_ratio:end)'; % dt_ratio = 10 -> iTg(1:10:401)' [41x1]
100     theta = A\b;
101
102     % Calculate parameter values and print
103     k = theta(1);
104     J1 = theta(2)*V*Cp; J2 = theta(3)*V*Cp; J3 = theta(4)*V*Cp;
105     fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J1, J2, J3);
106
107     % Forward simulate and store residuals
108     % Analytical
```

```

109 T_fwd = exp(-k*t_HR).*(T0+cumtrapz(t_HR, exp(k*t_HR).* ...
110     ((phi * theta(2:end)) + (k*T_amb))'));
111
112 % Picard: Error stepping, doesnt work
113 % T_fwd = T0 + cumtrapz(t_HR, -k*(T_fwd-T_amb) + (phi*theta(2:end))');
114 % % Works but doesnt make sense
115 % for jj = 2:N_HR % 2:401
116 %     T_fwd(1:jj) = T0 + cumtrapz(t_HR(1:jj), -k*(T_fwd(1:jj)-T_amb) + (phi(1:✓
117 %     jj, :)*theta(2:end))');
118 % end
119
120 % Time-stepping: Euler 1st order
121 % T_fwd(1) = T0;
122 % for jj = 2:N_HR % 2:401
123 %     T_fwd(jj) = T_fwd(jj-1) + (-k*(T_fwd(jj-1)-T_amb) + (phi(jj-1, :)*theta(2:✓
124 %     end)))*dt_HR;
125 % end
126
127 % Store residuals
128 psi(ii) = norm(T_fwd(1:dt_ratio:end)-T_data);
129
130 % Plot forward simulation every 5th iteration
131 if mod(ii, 5) == 0
132     plot(h1, t_HR, T_fwd, 'r:');ylim(h1, [20, 45]);
133     ylabel(h1, 'Temp (^oC)'); xlabel(h1, 'Time (s)');
134 end
135
136 % Plot Residuals
137 plot(h2, ii, psi(ii), 'ob','markerfacecolor','b');
138 ylabel(h2, '\bfresiduals \it\psi'); xlabel(h2, 'Iterations');
139
140 % Plot paramater convergence
141 yyaxis left; % J1, J2, J3 on left axis
142 plot(h3, ii, J1/1000, '.b', ii, J2/1000, '.g', ii, J3/1000, '.r');
143 ylabel(h3, 'J_x (kJ)'); xlabel(h3, 'Iterations'); ylim(h3, [0, 600]);
144
145 yyaxis right; % k on right axis
146 plot(h3, ii, k, '.k');
147 ylabel(h3, 'k (s^-1)'); xlabel(h3, 'Iterations'); ylim(h3, [0, 5]);
148 end
149
150 plot(h1, t_HR, T_fwd, 'b:');ylim(h1, [20, 45]);
151 ylabel(h1, 'Temp (^oC)'); xlabel(h1, 'Time (s)');
152
153 %% Q3 ARX
154 tutorial_setup
155
156 % Part a
157
158 [~, n_data] = size(T_data);
159
160 b = T_data(2:end)';

```

```
161 A = zeros(n_data-1, 5);
162 A(:, 1) = T_data(1:end-1)';
163 A(:, 5) = 1;
164
165 for c = 2:1:4
166     for ii = 1:1:n_data
167         if t_impulse(c-1) <= t_meas(ii)
168             A(ii-1, c) = 1;
169             break
170         end
171     end
172 end
173
174 theta = A\b;
175 spy(A)
176
177 T_arx1 = zeros(size(T_data));
178 T_arx1(1) = T_data(1);
179 for t = 2:n_data
180     T_arx1(t) = theta(1) * T_arx1(t-1) + theta(2:4)' * A(t-1, 2:4)' + theta(5);
181 end
182
183 figure(302);
184 plot(t_meas, T_arx1, 'r', t_meas, T_data, '+k');
185 ylim([20, 45]); xlabel('Time [s]'); ylabel('Temperature [^oC]');
186 hold on;
187
188 % Part b
189 t_s = 0:0.25:3;
190 [~, rows] = size(t_s);
191 phi = A(:, 2) + 2*A(:, 3) + 1.5*A(:, 4); % Combine inputs into one fn
192 A2 = zeros(rows, 3);
193 A2(:, 1) = T_data(1:rows);
194 A2(:, 2) = A(1:rows, 2); A2(end, 2) = 2;
195 A2(:, 3) = 1;
196 b2 = T_data(2:rows+1)';
197
198 theta2 = A2\b2; % Used for simulating, not param ID
199
200 T_arx2 = zeros(size(T_data));
201 T_arx2(1) = T_data(1);
202
203 for t = 2:n_data
204     T_arx2(t) = theta2(1)*T_arx2(t-1) + theta2(2)*phi(t-1) + theta2(3);
205 end
206
207 plot(t_meas, T_arx2, 'b');
208
209 %% Q4 Gradient Descent
210
211 tutorial_setup
212
213 % Forward simulation function
214 Temp_f = @(ts, t0, th, ph) exp(-th(1)*ts) .* (t0+cumtrapz(ts, ...
```

```

215     exp(th(1)*ts).*((ph*th(2:end))+(th(1)*T_amb))');
216
217 % Initializations
218 [~, n_data] = size(T_data);
219 J = zeros(n_data, cols);
220 n_iterations = 30;
221 T0 = mean(T_data(1:2));
222
223 dt_HR = 0.025;
224 t_HR = 0:dt_HR:10;
225 n_HR = 10/dt_HR;
226 phi = zeros(size(t_HR'));
227
228 for c = 1:1:3
229     for ii = 1:1:n_HR
230         if t_impulse(c) <= t_HR(ii)
231             phi(ii, c) = 1/dt_HR;
232             break
233         end
234     end
235 end
236
237 theta = zeros(cols, n_iterations);
238 psi = zeros(rows, 1);
239 TT = 0:dt_data/dt_HR:n_HR;
240
241 % Initial guess
242 delta = [0.05, 100, 10, 10]'; % Pertubations
243 lambda = 1;
244
245 % Guess theta(1), calculate psi(1)
246 theta(:, 1) = [0 0 0 0]'; % Initial conditions/guess
247 T_fwd = Temp_f(t_HR, T0, theta(:, 1), phi);
248 psi(:, 1) = T_fwd(TT+1)' - T_data';
249
250 for it = 1:n_iterations-1
251     % Calculate J: same for all Gauss-Newton, Levenberg, Levenberg-Marquardt
252     for col = 1:4
253         dtheta = zeros(4,1); dtheta(col, 1) = delta(col, 1);
254         dpsidth_i = (Temp_f(t_HR, T0, theta(:, it) + dtheta, phi) - T_fwd)/delta
255         J(:, col) = dpsidth_i(TT+1)';
256     end
257     % Next part is different for next iteration of theta!
258     % Levenberg
259     % theta(:, it+1) = theta(:, it) - (((J'*J)+lambda*eye(4))^-1) * (J'*psi(:,
260     it));
261     % T_fwd = Temp_f(t_HR, T0, theta(:, it+1), phi);
262     % psi(:, it+1) = T_fwd(TT+1)' - T_data';
263
264     % Gauss-Newton
265     % theta(:, it+1) = theta(:, it) - ((J'*J)^-1)*(J'*psi(:, it));
266     % T_fwd = Temp_f(t_HR, T0, theta(:, it+1), phi);
267     % psi(:, it+1) = T_fwd(TT+1)' - T_data';

```

```

267
268 % Levenberg-Marquardt
269 while true
270     % Calculate theta(ii+1)
271     theta(:, it+1) = theta(:, it) - (((J'*J)+lambda*eye(4).*(J'*J))^-1) * (J'*psi(:, it));
272
273     % Forward simulate for psi(ii+1)
274     T_fwd = Temp_f(t_HR, T0, theta(:, it+1), phi);
275     psi(:, it+1) = T_fwd(TT+1)' - T_data';
276
277     % Levenberg-Marquardt
278     % Update lambda, repeat if norm(psi(ii+1)) > norm(psi(ii))
279     if norm(psi(:, it+1)) <= norm(psi(:, it))
280         lambda = lambda / 5;
281         break
282     else
283         lambda = lambda * 5;
284     end
285 end
286
287 % Calculate the Paramaters to ID
288 k = theta(1, it+1);
289 Jx = theta(2:end, it+1)*V*Cp;
290 fprintf('GN: k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, Jx(1), Jx(2), Jx(3));
291 end
292
293 fprintf('\n');
294
295 % Steepest Descent
296 n_iterationsSD = 20000; % Takes a real fucking long time
297 thetaSD = zeros(cols, n_iterationsSD); thetaSD(:, 1) = [10 1 1 1]';
298 deltaSD = theta(:, end)/1000;
299 JSD = zeros(cols, 1);
300 alpha = 0.01;
301
302 for it = 1:n_iterationsSD-1
303     T_fwdSD = Temp_f(t_HR, T0, thetaSD(:, it), phi);
304     psi_thi = norm(T_fwdSD(TT+1)' - T_data');
305
306     for ii = 1:4
307         dtheta = zeros(4,1); dtheta(ii, 1) = deltaSD(ii, 1);
308         T_fwdSD_i = Temp_f(t_HR, T0, thetaSD(:, it) + dtheta, phi);
309         [T_fwdSD(TT+1)' T_fwdSD_i(TT+1)' T_data'];
310         psi_thi_i = norm(T_fwdSD_i(TT+1)' - T_data');
311         dpsi_dth_i = (psi_thi_i - psi_thi)/deltaSD(ii);
312         [dpsi_dth_i psi_thi_i psi_thi];
313         JSD(ii, 1) = dpsi_dth_i; % [4x1]
314     end
315     JSD;
316
317     thetaSD(:, it+1) = thetaSD(:, it) - alpha*JSD;
318     k = thetaSD(1, it+1);
319     Jx = thetaSD(2:end, it+1)*V*Cp;

```

```

320 fprintf('SD: k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, Jx(1), Jx(2), Jx(3));
321 end
322
323 figure(403);
324 yyaxis left;
325 plot([0 n_iterationsSD], [parent(2) parent(2)]/1000, 'k', ...
326      [0 n_iterationsSD], [parent(3) parent(3)]/1000, 'k', ...
327      [0 n_iterationsSD], [parent(4) parent(4)]/1000, 'k');
328 hold on
329 plot(1:n_iterationsSD, thetaSD(2, :)*V*Cp/1000, '.b', ...
330      1:n_iterationsSD, thetaSD(3, :)*V*Cp/1000, '.b', ...
331      1:n_iterationsSD, thetaSD(4, :)*V*Cp/1000, '.b');
332 ylim([0, 550]);
333 xlabel('Iterations');ylabel('J_x [kJ]')
334 hold all;
335
336 yyaxis right;
337 plot(1:n_iterationsSD, thetaSD(1, :), '.r');
338 plot([0 n_iterationsSD], [parent(1) parent(1)], 'r-');
339 xlabel('Iterations');ylabel('k [s^-1]')
340 ylim([0, 4]);
341
342 figure(401);
343 yyaxis left;
344 plot([0 n_iterations], [parent(2) parent(2)]/1000, 'k', ...
345      [0 n_iterations], [parent(3) parent(3)]/1000, 'k', ...
346      [0 n_iterations], [parent(4) parent(4)]/1000, 'k');
347 hold on
348 plot(1:n_iterations, theta(2, :)*V*Cp/1000, '.-b', ...
349      1:n_iterations, theta(3, :)*V*Cp/1000, '.b', ...
350      1:n_iterations, theta(4, :)*V*Cp/1000, ':b');
351 ylim([0, 550]);
352 xlabel('Iterations');ylabel('J_x [kJ]')
353 hold all;
354
355 yyaxis right;
356 plot(1:n_iterations, theta(1, :), '.r');
357 plot([0 n_iterations], [parent(1) parent(1)], 'r-');
358 xlabel('Iterations');ylabel('k [s^-1]')
359 ylim([0, 4]);
360
361 figure(402);
362 plot(t_HR, T_fwd, 'ob', t_HR, T_fwdSD, '--r'); ylim([20, 45]);
363 ylabel('Temp (^oC)'); xlabel('Time (s)');
364
365
366 %% Q5 Genetic Algorithm
367
368 tutorial_setup
369
370 % Initialize forward simulation variables
371 T0 = mean(T_data(1:2));
372 dt_ratio = 25;
373 dt_HR = dt_data/dt_ratio;

```

```
374 t_HR = t_start:dt_HR:t_end;
375 n_HR = t_end/dt_HR;
376 phi = zeros(size(t_HR'));
377
378 for c = 1:1:3
379     for ii = 1:1:n_HR
380         if t_impulse(c) <= t_HR(ii)
381             phi(ii, c) = 1/dt_HR;
382             break
383         end
384     end
385 end
386
387 % Forward simulation function
388 Temp_f = @(ts, t0, th, ph) exp(-th(1)*ts) .* (t0+cumtrapz(ts, ...
389     exp(th(1)*ts).*((ph*th(2:end))+(th(1)*T_amb))'));
390
391 % Initialize
392 generations = 250;
393 K = 100;
394 N = 10;
395 mut_rate = 0.05;
396 thetas = 30*rand(K, 5);
397 max_err = zeros(1, generations);
398
399 % Create plots
400 figure(501);
401
402 % K plot
403 ax_k = subplot(231); hold all;
404 xlabel(ax_k, 'Generations'); ylabel(ax_k, '\Theta_{opt}(k) [s^{-1}]');
405 plot(ax_k, [0 generations], [parent(1) parent(1)], '-g');
406 xlim(ax_k, [0, generations]); ylim(ax_k, [0 6]);
407
408 % J1, J2, J3 plot
409 ax_j = subplot(232); hold all;
410 xlabel(ax_j, 'Generations'); ylabel(ax_j, '\Theta_{opt}(J_x) [kJ]');
411 plot(ax_j, [0 generations], [parent(2) parent(2)]/1000, '-.r');
412 plot(ax_j, [0 generations], [parent(3) parent(3)]/1000, '-.b');
413 plot(ax_j, [0 generations], [parent(4) parent(4)]/1000, '-.m');
414 xlim(ax_j, [0, generations]); ylim(ax_j, [0 600]);
415
416 % Residuals plot
417 ax_ps = subplot(233); hold all;
418 xlabel(ax_ps, 'Generations'); ylabel(ax_ps, '\Psi_{opt}');
419 xlim(ax_k, [0, generations]);
420
421 % Forward simulation plot with theta opt
422 ax_sim = subplot(212); hold all;
423 xlabel(ax_sim, 'Time [s]'); ylabel(ax_sim, 'Temperature [^oC]');
424 ylim(ax_sim, [20 45]);
425
426 % Loop for each generation
427 for gen = 1:generations
```



```

428
429 % Iterate through parameter sets
430 for ii = 1:K
431     % Determine psi of theta(:, i)
432     theta = thetas(ii, 2:end)';
433     T_fwd = Temp_f(t_HR, T0, theta, phi);
434     thetas(ii, 1) = norm(T_fwd(1:dt_ratio:end)' - T_data');
435 end
436
437 % Sort thetas by the lowest (best) residuals
438 thetas = sortrows(thetas);
439
440 % Check for NaNs
441 for ii = 1:K
442     if isnan(thetas(ii, 1)) || isnan(thetas(ii, 1))
443         % Create an iterator of length K for only "good" organisms
444         ind=1; jj=1; iterator = zeros(1, K);
445         while jj <= 100
446             iterator(jj) = ind;
447             jj = jj + 1;
448             if mod(jj, ceil(K/ii)) == 0
449                 ind=ind+1;
450             end
451         end
452
453         % Appropriately remove NaN and Inf sets, replace with better clones
454         thetas(:, :) = thetas(iterator, :);
455         break
456     end
457 end
458
459 % Discard the worst and clone multiple copies of the top ranking sets
460 thetas(:, :) = thetas(ceil(((1:K)/20).^2), :);
461
462 % Mutate, keep one copy of
463 thetas(2:end, 2:end) = thetas(2:end, 2:end) + ...
464     mut_rate*[thetas(2:end,1) thetas(2:end,1) thetas(2:end,1) thetas(2:end,1)].*
465     *randn(K-1,4);
466
467 % Plot of theta opt
468 if mod(gen, 10) == 1
469     T_fwd = Temp_f(t_HR, T0, thetas(1, 2:end)', phi);
470     plot(ax_sim, t_HR, T_fwd, 'r');
471 end
472
473 % Plot optimal values from this generation
474 ps = thetas (1,1);
475
476 k = thetas(1,2); J = thetas(1,3:end)*V*Cp;
477 fprintf('k=%.2f, J1=%.0f, J2=%.0f, J3=%.0f\n', k, J(1), J(2), J(3));
478
479 plot(ax_k, gen, k, '.g');
480 plot(ax_j, gen, J(1)/1000, '.r', gen, J(2)/1000, '.b', gen, J(3)/1000, '.m');
481 plot(ax_ps, gen, ps, '.k');

```

```
481 end
482
483
484 plot(ax_sim, t_HR, T_fwd, 'k');
485 plot(ax_sim, t_meas, T_data, '+b');
486
```