

1. (a)  $\alpha^k = \arg \min f(x^k + \alpha d^k)$  with  $d^k = -\nabla f(x^k)$

Since  $\alpha^k$  minimizes  $h(\alpha) = f(x^k + \alpha d^k)$ ,  $h'(\alpha)$  must be 0 at  $\alpha = \alpha^k$

Then, we have  $h'(\alpha) = \frac{d f(x^k + \alpha d^k)}{d \alpha} = \langle d^k, \nabla f(x^k + \alpha d^k) \rangle$

Thus,  $h'(\alpha^k) = \langle d^k, \nabla f(x^k + \alpha^k d^k) \rangle = 0$

(b) Since  $d^k = -\nabla f(x^k)$ ,

$$\langle d^k, \nabla f(x^k + \alpha^k d^k) \rangle = \langle -\nabla f(x^k), \nabla f(x^k + \alpha^k d^k) \rangle = 0$$

$x_k$  and  $x_{k+1}$  are two consecutive points generated by the steepest descent algorithm, then we have

$$x^{k+1} = x^k + \alpha^k d^k$$

Thus,  $\langle -\nabla f(x^k), \nabla f(x^{k+1}) \rangle = 0$

$$\langle \nabla f(x^k), \nabla f(x^{k+1}) \rangle = 0$$

---

```

% Optimization Homework3 Problem2

% Problem2(a)

% Define the function f
f = @(x1, x2) x1.^2 + 5*x2.^2 + 4*x1.*x2 - 6*x1 - 14*x2 + 20;

% Define the gradient of f in terms of x1 and x2
grad_f = @(x1, x2) [2*x1 + 4*x2 - 6; 10*x2 + 4*x1 - 14];

% Initial value
x = [10; 10];
% Tolerance
epsilon = 1e-6;
% Counter of iterations
iterations = 0;

% Store the values during iterations
results = [];
% Store the path of iterations
iterate_history = x';

while norm(grad_f(x(1), x(2))) > epsilon

    % Compute the gradient at current point x=(x1, x2)
    d_k = -grad_f(x(1), x(2));

    % According to Problem1
    % Minimize f(x_k + alpha * d_k) with respect to alpha
    % Symbol variable declaration
    syms alpha_sym;
    % Establish the function f_alpha = f(x_k + alpha * d_k)
    f_alpha = f(x(1) + alpha_sym * d_k(1), x(2) + alpha_sym * d_k(2));
    % Calculate the derivative of f_alpha with respect to alpha
    df_alpha = diff(f_alpha, alpha_sym);
    % Find optimal alpha when df_alpha=0
    alpha_k = double(solve(df_alpha == 0, alpha_sym));

    % Update x
    x = x + alpha_k * d_k;

    % Store results
    results = [results; x(1), x(2), d_k(1), d_k(2), norm(d_k), alpha_k,
    f(x(1), x(2))];

    % Store the path for plotting
    iterate_history = [iterate_history; x'];
    % Iteration counter
    iterations = iterations + 1;

end

% Create a table with the results

```

---

---

```

result_table = array2table(results, 'VariableNames', {'xk1', 'xk2', 'dk1',
'dk2', 'Norm_dk', 'alpha_k', 'f_xk'});

% Display the table
disp(result_table);

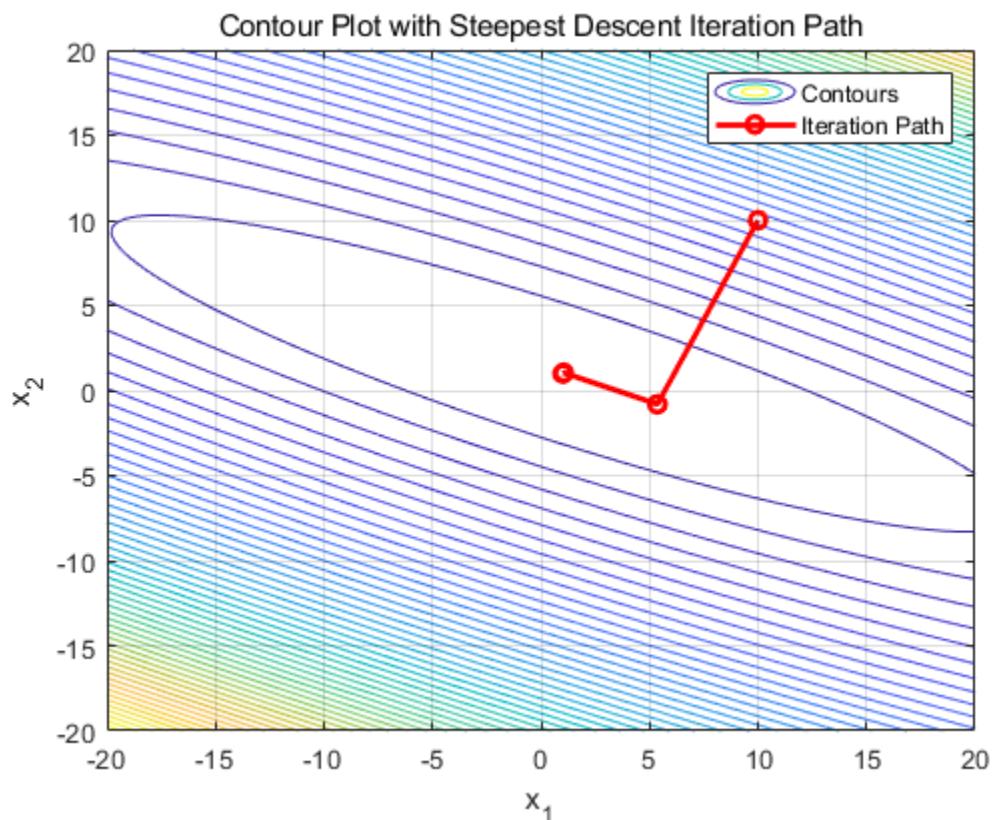
% Plot the contour of the function and the iteration path
[x1_vals, x2_vals] = meshgrid(-20:0.1:20, -20:0.1:20);
f_vals = f(x1_vals, x2_vals);

figure;
contour(x1_vals, x2_vals, f_vals, 50);
hold on;
plot(iterate_history(:, 1), iterate_history(:, 2), '-ro', 'LineWidth', 2,
'MarkerSize', 6);
xlabel('x_1');
ylabel('x_2');
title('Contour Plot with Steepest Descent Iteration Path');
grid on;
legend('Contours', 'Iteration Path');



| xk1      | xk2      | dk1         | dk2         | Norm_dk    |
|----------|----------|-------------|-------------|------------|
| alpha_k  | f_xk     |             |             |            |
| 5.3669   | -0.81065 | -54         | -126        | 137.08     |
| 0.085799 | 13.834   |             |             |            |
| 1.0426   | 1.0426   | -1.4911     | 0.63905     | 1.6223     |
| 2.9      | 10.018   |             |             |            |
| 1.0207   | 0.99143  | -0.25562    | -0.59645    | 0.64892    |
| 0.085799 | 10       |             |             |            |
| 1.0002   | 1.0002   | -0.0070586  | 0.0030251   | 0.0076795  |
| 2.9      | 10       |             |             |            |
| 1.0001   | 0.99996  | -0.00121    | -0.0028234  | 0.0030718  |
| 0.085799 | 10       |             |             |            |
| 1        | 1        | -3.3413e-05 | 1.432e-05   | 3.6353e-05 |
| 2.9      | 10       |             |             |            |
| 1        | 1        | -5.728e-06  | -1.3365e-05 | 1.4541e-05 |
| 0.085799 | 10       |             |             |            |


```



Published with MATLAB® R2023b

---

```

% Clear workspace and command window
clear; clc;

% Define the function f
f = @(x) x(1)^2 + 5*x(2)^2 + 4*x(1)*x(2) - 6*x(1) - 14*x(2) + 20;

% Define the initial point same as problem(a)
x0 = [10; 10];

% Set options for fminunc
opt = optimoptions(@fminunc, ...
    'Display', 'iter-detailed', ...
    'Algorithm', 'quasi-newton', ...
    'HessUpdate', 'steepdesc', ...
    'MaxFunctionEvaluations', 2000);

% Call the optimization solver fminunc
[x, fval, exitflag, output] = fminunc(f, x0, opt);

% Display the results
disp('Optimization Results:');
disp(['Optimal point x: ', mat2str(x)]);
disp(['Function value at optimal point: ', num2str(fval)]);
disp(['Number of iterations: ', num2str(output.iterations)]);
disp(['Exit flag: ', num2str(exitflag)]);

```

<i>Iteration</i>	<i>Func-count</i>	<i>f(x)</i>	<i>Step-size</i>	<i>First-order optimality</i>
0	3	820		126
1	9	18.3673	0.0793651	8.86
2	15	13.7159	0.1	2.17
3	21	13.3007	0.155068	1.9
4	27	12.9319	0.180141	1.93
5	33	12.6043	0.155068	1.69
6	39	12.3133	0.180141	1.71
7	45	12.0548	0.155068	1.5
8	51	11.8252	0.180141	1.52
9	57	11.6213	0.155068	1.33
10	63	11.4401	0.180141	1.35
11	69	11.2792	0.155068	1.18
12	75	11.1363	0.180141	1.2
13	81	11.0093	0.155068	1.05
14	87	10.8965	0.180141	1.07
15	93	10.7964	0.155068	0.933
16	99	10.7074	0.18014	0.947
17	105	10.6283	0.155068	0.828
18	111	10.5581	0.18014	0.842
19	117	10.4958	0.155068	0.736
<i>First-order optimality</i>				
<i>Iteration</i>	<i>Func-count</i>	<i>f(x)</i>	<i>Step-size</i>	<i>First-order optimality</i>
20	123	10.4404	0.18014	0.748
21	129	10.3912	0.155068	0.654
22	135	10.3475	0.18014	0.664

---

				<i>First-order</i>
				optimality
23	141	10.3086	0.155068	0.581
24	147	10.2742	0.18014	0.59
25	153	10.2435	0.155068	0.516
26	159	10.2163	0.18014	0.524
27	165	10.1921	0.155069	0.458
28	171	10.1707	0.18014	0.465
29	177	10.1516	0.155069	0.407
30	183	10.1347	0.18014	0.413
31	189	10.1196	0.155069	0.361
32	195	10.1062	0.18014	0.367
33	201	10.0944	0.155069	0.321
34	207	10.0838	0.18014	0.326
35	213	10.0745	0.155069	0.285
36	219	10.0661	0.18014	0.29
37	225	10.0588	0.155069	0.253
38	231	10.0522	0.18014	0.257
39	237	10.0464	0.155069	0.225
				<i>First-order</i>
Iteration	Func-count	f(x)	Step-size	optimality
40	243	10.0412	0.18014	0.229
41	249	10.0366	0.155069	0.2
42	255	10.0325	0.180139	0.203
43	261	10.0289	0.155069	0.178
44	267	10.0256	0.180139	0.18
45	273	10.0228	0.155069	0.158
46	279	10.0202	0.180139	0.16
47	285	10.018	0.15507	0.14
48	291	10.016	0.180139	0.142
49	297	10.0142	0.15507	0.124
50	303	10.0126	0.180138	0.126
51	309	10.0112	0.15507	0.111
52	315	10.0099	0.180138	0.112
53	321	10.0088	0.15507	0.0982
54	327	10.0078	0.180138	0.0997
55	333	10.007	0.15507	0.0872
56	339	10.0062	0.180137	0.0886
57	345	10.0055	0.15507	0.0775
58	351	10.0049	0.180137	0.0787
59	357	10.0043	0.155071	0.0688
				<i>First-order</i>
Iteration	Func-count	f(x)	Step-size	optimality
60	363	10.0039	0.180137	0.0699
61	369	10.0034	0.15507	0.0611
62	375	10.003	0.180136	0.0621
63	381	10.0027	0.155071	0.0543
64	387	10.0024	0.180137	0.0551
65	393	10.0021	0.155071	0.0482
66	399	10.0019	0.180136	0.049
67	405	10.0017	0.155071	0.0428
68	411	10.0015	0.180136	0.0435
69	417	10.0013	0.155071	0.0381
70	423	10.0012	0.180135	0.0387
71	429	10.001	0.155072	0.0338
72	435	10.0009	0.180135	0.0343

---

---

73	441	10.0008	0.155072	0.03
74	447	10.0007	0.180133	0.0305
75	453	10.0007	0.155073	0.0267
76	459	10.0006	0.180135	0.0271
77	465	10.0005	0.155073	0.0237
78	471	10.0005	0.180136	0.0241
79	477	10.0004	0.15507	0.021
<i>First-order optimality</i>				
Iteration	Func-count	f(x)	Step-size	optimality
80	483	10.0004	0.180136	0.0214
81	489	10.0003	0.155073	0.0187
82	495	10.0003	0.180134	0.019
83	501	10.0003	0.155074	0.0166
84	507	10.0002	0.180129	0.0169
85	513	10.0002	0.155077	0.0147
86	519	10.0002	0.18013	0.015
87	525	10.0002	0.155081	0.0131
88	531	10.0001	0.180125	0.0133
89	537	10.0001	0.155081	0.0116
90	543	10.0001	0.180122	0.0118
91	549	10.0001	0.155084	0.0103
92	555	10.0001	0.18012	0.0105
93	561	10.0001	0.155087	0.00918
94	567	10.0001	0.180114	0.00932
95	573	10.0001	0.155088	0.00815
96	579	10.0001	0.180114	0.00828
97	585	10	0.155086	0.00724
98	591	10	0.180115	0.00736
99	597	10	0.155087	0.00643
<i>First-order optimality</i>				
Iteration	Func-count	f(x)	Step-size	optimality
100	603	10	0.180116	0.00654
101	609	10	0.155086	0.00572
102	615	10	0.180112	0.00581
103	621	10	0.155096	0.00508
104	627	10	0.180106	0.00516
105	633	10	0.155092	0.00451
106	639	10	0.1801	0.00458
107	645	10	0.1551	0.00401
108	651	10	0.180105	0.00407
109	657	10	0.155107	0.00356
110	663	10	0.180062	0.00361
111	669	10	0.155115	0.00316
112	675	10	0.180085	0.00321
113	681	10	0.155109	0.00281
114	687	10	0.180073	0.00285
115	693	10	0.155111	0.00249
116	699	10	0.180067	0.00253
117	705	10	0.155132	0.00222
118	711	10	0.180063	0.00225
119	717	10	0.155146	0.00197
<i>First-order optimality</i>				
Iteration	Func-count	f(x)	Step-size	optimality
120	723	10	0.180036	0.002

---

---

121	729	10	0.155162	0.00175
122	735	10	0.179974	0.00177
123	741	10	0.155165	0.00155
124	747	10	0.179986	0.00158
125	753	10	0.155204	0.00138
126	759	10	0.179974	0.0014
127	765	10	0.155198	0.00123
128	771	10	0.179911	0.00124
129	777	10	0.155219	0.00109
130	783	10	0.17995	0.0011
131	789	10	0.155249	0.000967
132	795	10	0.17995	0.000981
133	801	10	0.155226	0.000859
134	807	10	0.179886	0.000871
135	813	10	0.155253	0.000763
136	819	10	0.179829	0.000774
137	825	10	0.155273	0.000678
138	831	10	0.17989	0.000687
139	837	10	0.155264	0.000602
<i>First-order</i>				
Iteration	Func-count	f(x)	Step-size	optimality
140	843	10	0.17993	0.000611
141	849	10	0.155264	0.000535
142	855	10	0.179837	0.000542
143	861	10	0.155295	0.000475
144	867	10	0.17983	0.000482
145	873	10	0.155347	0.000422
146	879	10	0.179768	0.000428
147	885	10	0.155473	0.000375
148	891	10	0.179601	0.00038
149	897	10	0.155599	0.000333
150	903	10	0.179388	0.000337
151	909	10	0.155645	0.000296
152	915	10	0.179318	0.0003
153	921	10	0.155552	0.000263
154	927	10	0.17946	0.000266
155	933	10	0.155582	0.000234
156	939	10	0.179444	0.000237
157	945	10	0.155499	0.000207
158	951	10	0.17942	0.00021
159	957	10	0.155966	0.000184
<i>First-order</i>				
Iteration	Func-count	f(x)	Step-size	optimality
160	963	10	0.178875	0.000186
161	969	10	0.156208	0.000164
162	975	10	0.178814	0.000165
163	981	10	0.156522	0.000146
164	987	10	0.178024	0.000147
165	993	10	0.156517	0.00013
166	999	10	0.178039	0.00013
167	1005	10	0.156769	0.000115

□化已完成：一口最口性口度 9.065199e-07 小于  
options.OptimalityTolerance = 1.000000e-06。

---

*Optimization Results:*

*Optimal point x: [1.00024444528461; 0.999906538828552]*

*Function value at optimal point: 10*

*Number of iterations: 167*

*Exit flag: 1*

*Published with MATLAB® R2023b*

$$3. (a) f(x) = \frac{x_1^4}{4} - x_1^2 + 2x_1 + (x_2 - 1)^2$$

$$(\nabla f)(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} x_1^3 - 2x_1 + 2 \\ 2(x_2 - 1) \end{pmatrix}$$

$$(Hf)(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 3x_1^2 - 2 & 0 \\ 0 & 2 \end{pmatrix}$$

According to Newton's method,  $x^{k+1} = x^k - (Hf)(x^k)^{-1}(\nabla f)(x^k)$

For  $k=0$ , we have  $x_0 = \begin{pmatrix} x_{0,1} \\ x_{0,2} \end{pmatrix}$

$$(Hf)(x_0) = \begin{pmatrix} 3x_{0,1}^2 - 2 & 0 \\ 0 & 2 \end{pmatrix}$$

According to Newton's method, we need  $(Hf)(x^k)$  to be invertible

at each iteration.  $Hf(x_0)$  is invertible iff  $\det((Hf)(x_0)) \neq 0$

$$\text{Thus, } \det((Hf)(x_0)) = 6x_{0,1}^2 - 4 = 0$$

$$\Rightarrow x_{0,1} = \pm \sqrt{\frac{2}{3}}$$

Therefore, the initialization value  $x_0 = \begin{pmatrix} \pm \sqrt{\frac{2}{3}} \\ x_{0,2} \end{pmatrix}$ ,  $x_{0,2} \in \mathbb{R}$  are points

where  $(Hf)(x_0)$  is non-invertible and the Newton's method breaks down

at these points.

---

```
% Homework3 Problem3(b)
% Newton's method for minimizing the function f

% Clear workspace and command window
clear; clc;

% Define the function f as an anonymous function
f = @(x) (x(1).^4)/4 - x(1).^2 + 2*x(1) + (x(2) - 1).^2;

% Define the gradient of f as an anonymous function
grad_f = @(x) [ x(1).^3 - 2*x(1) + 2 ; 2*(x(2) - 1) ];

% Define the Hessian matrix of f as an anonymous function
Hessian_f = @(x) [ 3*x(1).^2 - 2, 0 ; 0, 2 ];

% Initial guess for the minimizer
x_initial = [2 ; 2]; % Starting point x0

% Set tolerance for the gradient norm
tolerance = 1e-6;

% Set maximum number of iterations
max_iterations = 100;

% Iteration counter
iteration = 0;

% Initialize a matrix to store all iterates for plotting
iterate_history = x_initial;

% Newton's method
while norm(grad_f(x_initial)) > tolerance && iteration < max_iterations

    % Compute the gradient at the current point
    gradient_current = grad_f(x_initial);

    % Compute the Hessian matrix at the current point
    Hessian_current = Hessian_f(x_initial);

    % Check if the Hessian is positive definite
    eigenvalues = eig(Hessian_current);
    if any(eigenvalues <= 0)
        error('Hessian is not positive definite at iteration %d', iteration);
    end

    % Update the current point
    x_initial = x_initial - inv(Hessian_current) * gradient_current;

    % Store the new iterate
    iterate_history = [iterate_history, x_initial];

    % Increment iteration counter
    iteration = iteration + 1;
end
```

---

---

```

iteration = iteration + 1;

end

% Display the number of iterations taken
fprintf('Number of iterations: %d\n', iteration);

% Plotting section

% Create a grid of values for x1 and x2 for contour plotting
x1_values = linspace(-3, 3, 100);
x2_values = linspace(-3, 5, 100);
[X1_grid, X2_grid] = meshgrid(x1_values, x2_values);

% Compute the function values over the grid for contour plotting
F_values = (X1_grid.^4)/4 - X1_grid.^2 + 2*X1_grid + (X2_grid - 1).^2;

% Generate contour plot of the function f
figure;
contour_levels = 50; % Number of contour levels
contour(X1_grid, X2_grid, F_values, contour_levels); % Plot contours
hold on; % Retain current plot when adding new plots

% Plot the path traced by the Newton's method iterations
plot(iterate_history(1,:), iterate_history(2,:), 'ro-', 'LineWidth', 2);

% Mark the starting point
plot(iterate_history(1,1), iterate_history(2,1), 'go', 'MarkerFaceColor',
'g', 'MarkerSize', 10);

% Mark the ending point (estimated minimum)
plot(iterate_history(1,end), iterate_history(2,end), 'bo',
'MarkerFaceColor', 'b', 'MarkerSize', 10);

% Add labels and title to the plot
xlabel('x_1');
ylabel('x_2');
title('Contour Plot of f and Path of Newton''s Method Iterations');

% Add legend to the plot
legend('Contours of f', 'Newton''s Method Path', 'Starting Point',
'Estimated Minimum');

% Release the hold on the current plot
hold off;

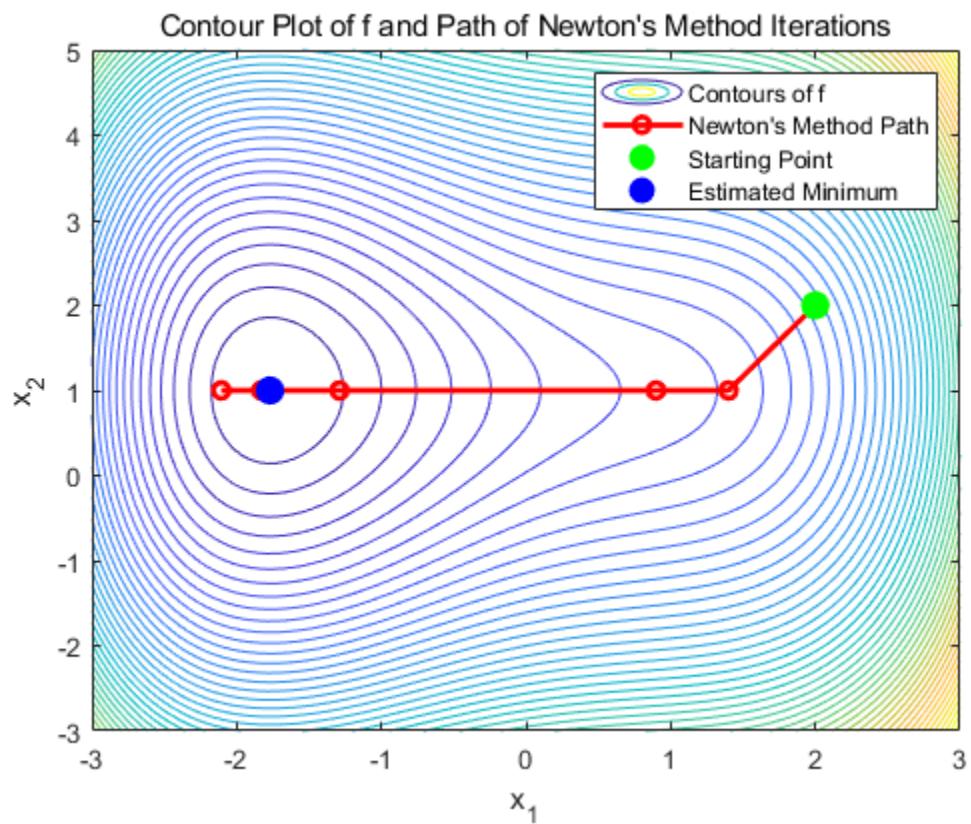
% Display the estimated minimum point and function value at that point
x_min = x_initial;
f_min = f(x_min);
fprintf('Estimated minimum at x = [%f, %f], f(x) = %f\n', x_min(1),
x_min(2), f_min);

```

*Number of iterations: 8*

*Estimated minimum at x = [-1.769292, 1.000000], f(x) = -4.219136*

---



Published with MATLAB® R2023b