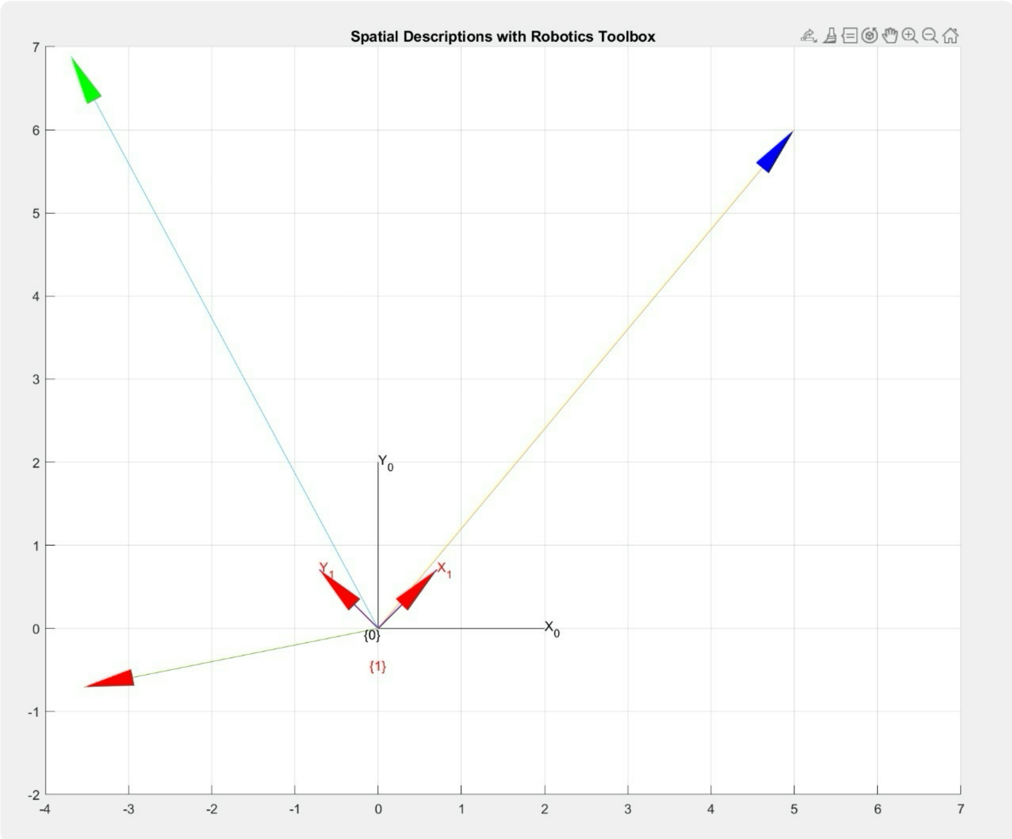
Answer Sheet

Index No: 220343B

1. MATLAB code for 3.1 ~ 3.5.

```
% Spatial Descriptions with Robotics Toolbox
clc; clear; close all;

figure; hold on; axis equal;
title('Spatial Descriptions with Robotics Toolbox');

% 3.1 Default frame {0}
trplot2(SE2(), 'frame', '0', 'color', 'k', 'length', 2);
axis([-4,7,-2,7])
grid on;
% 3.2 Point p = [5;6] in frame {0}
p_in_0 = [5; 6];
plot_arrow([0 0], p_in_0', 'b');

% 3.3 Frame {1} rotated 45° CCW from {0}
theta = deg2rad(45);
R1_in_0 = rot2(theta);
tranimate2(R1_in_0, 'frame', '1', 'color', 'r', 'arrow')

% Transform point p into frame {1}
p_in_1 = R1_in_0' * p_in_0;  % transformation
disp('p in frame {1}:');
disp(p_in_1);

% 3.4 Define q = [-3;2] in frame {1}
q_in_1 = [-3; 2];
q_in_0 = R1_in_0 * q_in_1;   % expressed in frame {0} for visualization
plot_arrow([0 0], q_in_0', 'r');

% 3.5 Rotate p by 68° CCW in {0}
phi = deg2rad(68);
Rphi = rot2(phi);
r_in_0 = Rphi * p_in_0;
plot_arrow([0 0], r_in_0', 'g');
```

2. Final output MATLAB figure for the operations in 3.1 ~ 3.5.



3. p^1 for 3.3:

```
p in frame {1}:
    7.7782
    0.7071
```

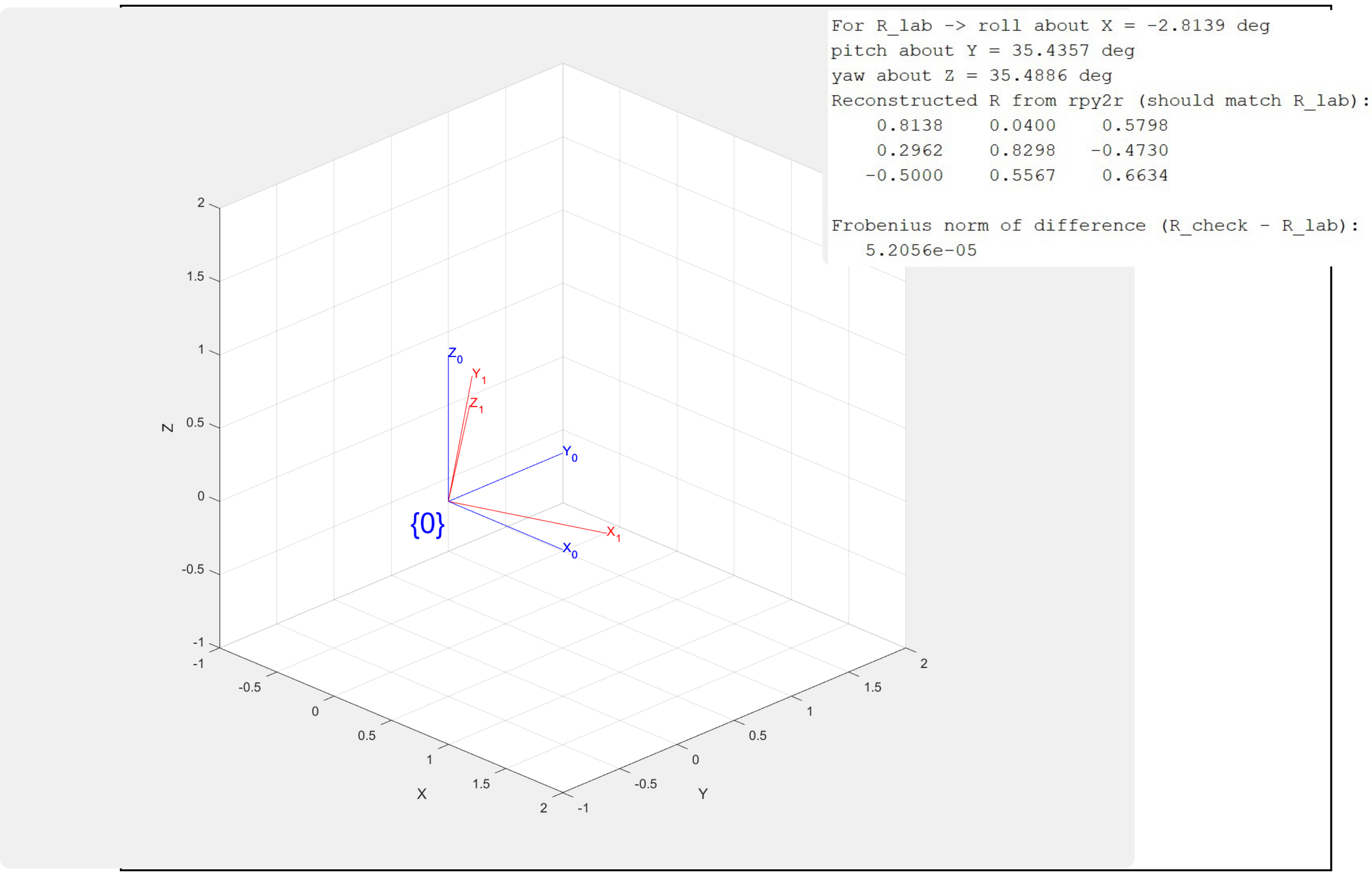
4. R_1^0 for 3.7.

```
R1_0 (rotation matrix from 3.7):
    0.7424   -0.5198    0.4226
    0.6436    0.7285   -0.2346
   -0.1859    0.4462    0.8754
```

5. MATLAB code for 3.6 ~ 3.9.

```
% 3.6 - 3.9: 3D frames, rotations, and RPY extraction
clc; clear; close all;

% 3.6: Visualize default 3D frame {0}
figure;
trplot(eye(4), 'frame', '0', 'color','b'); % default frame
axis([-1 2 -1 2 -1 2]);
grid on;
view(45,25);
hold on;

% 3.7: Build successive rotations (intrinsic rotations about current axes)

% Use rotx, roty, rotz (degrees)
R_x = rotx(deg2rad(15));            % rotation about X (degrees)
R_xy = R_x * roty(deg2rad(25));    % then about new Y
R_xyz = R_xy * rotz(deg2rad(35));  % then about new Z

% R1_0 is the rotation of frame {1} relative to {0} (3x3)
R1_0 = R_xyz;                  % 3x3 rotation matrix

% Visualize intermediate frames with animation (optional cleanup)
tranimate(eye(4), 'frame', '0', 'color','b');      % base
pause(0.5);
T1 = eye(4); T1(1:3,1:3) = R_x;     tranimate(T1, 'frame', 'after_X', 'color','k', 'cleanup', true);
pause(0.5);
T2 = eye(4); T2(1:3,1:3) = R_xy;    tranimate(T2, 'frame', 'after_XY','color','m','cleanup', true);
pause(0.5);
T3 = eye(4); T3(1:3,1:3) = R_xyz;   tranimate(T3, 'frame', '1', 'color','r');

% Display R1_0
disp('R1_0 (rotation matrix from 3.7):');
disp(R1_0);

% 3.9: Given rotation matrix R (from the lab), find psi (roll about X), theta (pitch about Y), phi (yaw about Z)
R_lab = [ 0.8138  0.0400  0.5798;
          0.2962  0.8298 -0.4730;
         -0.5000  0.5567  0.6634 ];

% Use tr2rpy with 'deg' and 'xyz' to request roll(X)-pitch(Y)-yaw(Z) ordering
rpy_deg = tr2rpy(R_lab, 'deg', 'xyz');  % returns [roll pitch yaw] in degrees
psi_deg   = rpy_deg(1); % roll about X
theta_deg = rpy_deg(2); % pitch about Y
phi_deg   = rpy_deg(3); % yaw about Z

fprintf('For R_lab -> roll about X = %.4f deg\n', psi_deg);
fprintf('pitch about Y = %.4f deg\n', theta_deg);
fprintf('yaw about Z = %.4f deg\n', phi_deg);

% Confirm by rebuilding R from these angles and comparing to R_lab
R_check = rpy2r(psi_deg, theta_deg, phi_deg, 'deg', 'xyz');
disp('Reconstructed R from rpy2r (should match R_lab):');
disp(R_check);
disp('Frobenius norm of difference (R_check - R_lab):');
disp(norm(R_check - R_lab, 'fro'));
```

6. Final output MATLAB figure for the operations in 3.6 ~ 3.9.



7. Default roll-pitch-yaw angle definition for the toolbox.

Default Roll-Pitch-Yaw order → ZYX (Yaw, Pitch, roll)

8. For 3.9,
ψ: -2.8139° θ: 35.4357° φ: 35.4886°