

HW 8 Solutions

Problem 5

You also needed to download CQuickReturn.chf and quickreturn.h from the link provided in the homework problem set for this program to run.

```
1  /**
2   * File: quick.ch
3   * Author: Nicolas Ventura
4   */
5
6  #include "quickreturn.h" // Include the header file
7
8  int main(void) {
9      CQuickReturn qr;
10     int numpoints = 100; // Number of frames to animate
11
12     bool unit = SI; // SI units are in meters
13     double r1 = 2.5/100,
14            r2 = 1.0/100,
15            r4 = 6.5/100,
16            r5 = 3.0/100,
17            r6,
18            r7 = 5.0/100;
19     double theta1 = deg2rad(90.0),
20            theta2 = deg2rad(30.0),
21            theta3, theta4;
22
23     qr.uscUnit(unit);
24     qr.setLinks(r1, r2, r4, r5, r7, theta1);
25     qr.setNumPoints(numpoints);
26
27     /* Problem 5 */
28     theta3 = qr.getAngPos(theta2, 3);
29     theta4 = qr.getAngPos(theta2, 4);
30     r6 = qr.sliderPos(theta2);
31     printf("Theta3 = %lf rad = %lf deg\n", theta3, rad2deg(theta3))
32     ;
33     printf("Theta4 = %lf rad = %lf deg\n", theta4, rad2deg(theta4))
34     ;
35     printf("r6 = %lf m\n", r6);
36     /* Part (a) */
37     qr.displayPosition(theta2, QANIMATE_OUTPUTTYPE_DISPLAY);
38     /* Part (b) */
39     qr.animation(QANIMATE_OUTPUTTYPE_DISPLAY);
40     return 0;
41 }
```

```
1  Theta3 = 1.289761 rad = 73.897886 deg
2  Theta4 = -0.427942 rad = -24.519274 deg
3  r6 = 0.045322 m
```

Problem 6

```

1  /**
2  *
3  * File: prob6.ch
4  * Author: Nicolas Ventura
5  * Run a kinematic analysis
6  * on a standard crank-rocker
7  * fourbar linkage.
8  */
9
10 #include <fourbar.h>
11
12 #define NUMPOINTS 100
13
14 int main(void) {
15     /// Create variables ///
16     CFourbar fourbar;
17     CPlot plot1, plot2, plotV;
18     int i;
19     double r[1:4], rp, beta, theta2[NUMPOINTS], t[NUMPOINTS];
20     /// SOLUTION 1 SOLUTION 2 ///
21     array double
22         theta_1[1:4],      theta_2[1:4],
23         omega_1[1:4],      omega_2[1:4],
24         alpha_1[1:4],      alpha_2[1:4],
25         V_1[NUMPOINTS],    V_2[NUMPOINTS],
26         theta3[NUMPOINTS], theta4[NUMPOINTS];
27
28     double
29         gamma_1,           gamma_2;
30     complex double
31         P_1,               P_2,
32         Vp_1,              Vp_2,
33         Ap_1,              Ap_2;
34
35     /// Define known values ///
36     r[1] = 5.0 / 100;
37     r[2] = 2.0 / 100;
38     r[3] = 3.0 / 100;
39     r[4] = 4.5 / 100;
40     rp = 2.5 / 100;
41     beta = deg2rad(30.0);
42     linspace(theta2, deg2rad(0.0), deg2rad(360.0));
43
44     /// Set knowns for first solution ///
45     theta_1[1] = deg2rad(15.0);
46     theta_1[2] = deg2rad(45.0);
47     omega_1[1] = 0.0; // rad/sec
48     omega_1[2] = 15.0; // rad/sec
49     alpha_1[1] = 0.0; // rad/s^2
50     alpha_1[2] = 0.0; // rad/s^2
51
52     /// Set knowns for second solution ///
53     theta_2[1] = deg2rad(15.0);
54     theta_2[2] = deg2rad(45.0);
55     omega_2[1] = 0.0; // rad/sec
56     omega_2[2] = 15.0; // rad/sec
57     alpha_2[1] = 0.0; // rad/s^2
58     alpha_2[2] = 0.0; // rad/s^2

```

```

58
59  //// Basic fourbar setup ////
60  fourbar.uscUnit(false);
61  fourbar.setLinks(r[1], r[2], r[3], r[4], theta_1[1]);
62  fourbar.setCouplerPoint(rp, beta);
63
64  //// Part (a) ////
65  fourbar.angularPos(theta_1, theta_2, FOURBAR_LINK2);
66  printf("Solution 1:\n  theta [rad] = %9.4lf", theta_1);
67  printf("Solution 2:\n  theta [rad] = %9.4lf\n", theta_2);
68
69  //// Part (b) ////
70  fourbar.couplerPointPos(theta_1[2], P_1, P_2);
71  printf("Solution 1:\n  P = (%9.4lf, %9.4lf) [m]\n", real(P_1),
72  imag(P_1));
73  printf("Solution 2:\n  P = (%9.4lf, %9.4lf) [m]\n\n", real(P_2),
74  imag(P_2));
75
76  //// Part (c) ////
77  fourbar.transAngle(gamma_1, gamma_2, theta_1[2], FOURBAR_LINK2);
78  ;
79  printf("Solution 1:\n  gamma = %9.4lf [rad] = %9.4lf [deg]\n",
80  gamma_1, rad2deg(gamma_1));
81  printf("Solution 2:\n  gamma = %9.4lf [rad] = %9.4lf [deg]\n\n",
82  gamma_2, rad2deg(gamma_2));
83
84  //// Part (d) ////
85  fourbar.angularVel(theta_1, omega_1, FOURBAR_LINK2);
86  fourbar.angularVel(theta_2, omega_2, FOURBAR_LINK2);
87  Vp_1 = fourbar.couplerPointVel(theta_1[2], theta_1[3], omega_1
88  [2], omega_1[3]);
89  Vp_2 = fourbar.couplerPointVel(theta_2[2], theta_2[3], omega_2
90  [2], omega_2[3]);
91  printf("Solution 1:\n  omega [rad/s] = %9.4lf  Vp = (%9.4lf,
92  %9.4lf) [m/s] = %9.4lf [m/s]\n", omega_1, real(Vp_1), imag(Vp_1),
93  cabs(Vp_1));
94  printf("Solution 2:\n  omega [rad/s] = %9.4lf  Vp = (%9.4lf,
95  %9.4lf) [m/s] = %9.4lf [m/s]\n\n", omega_2, real(Vp_2), imag(
96  Vp_2), cabs(Vp_2));
97
98  //// Part (e) ////
99  fourbar.angularAccel(theta_1, omega_1, alpha_1, FOURBAR_LINK2);
100  fourbar.angularAccel(theta_2, omega_2, alpha_2, FOURBAR_LINK2);
101  Ap_1 = fourbar.couplerPointAccel(theta_1[2], theta_1[3],
102  omega_1[2], omega_1[3], alpha_1[2], alpha_1[3]);
103  Ap_2 = fourbar.couplerPointAccel(theta_2[2], theta_2[3],
104  omega_2[2], omega_2[3], alpha_2[2], alpha_2[3]);
105  printf("Solution 1:\n  alpha [rad/s^2] = %9.4lf  Ap = (%9.4lf,
106  %9.4lf) [m/s^2] = %9.4lf [m/s^2]\n", alpha_1, real(Ap_1), imag(
107  Ap_1), cabs(Ap_1));
108  printf("Solution 2:\n  alpha [rad/s^2] = %9.4lf  Ap = (%9.4lf,
109  %9.4lf) [m/s^2] = %9.4lf [m/s^2]\n", alpha_2, real(Ap_2), imag(
110  Ap_2), cabs(Ap_2));
111
112  //// Part (f) ////
113  fourbar.plotAngularVels(&plot1, 1);
114  fourbar.plotAngularVels(&plot2, 2);

```

```

98 // Calculate the t array
99 fourbar.setAngularVel(theta_1[2]);
100 fourbar.angularPos(1, t, theta3, theta4);
101 for (i = 0; i < NUMPOINTS; i++) {
102 // Need to re-calculate for every theta2
103 theta_1[2] = theta2[i];
104 theta_2[2] = theta2[i];
105 fourbar.angularPos(theta_1, theta_2, FOURBAR_LINK2);
106 fourbar.angularVel(theta_1, omega_1, FOURBAR_LINK2);
107 fourbar.angularVel(theta_2, omega_2, FOURBAR_LINK2);
108 Vp_1 = fourbar.couplerPointVel(theta_1[2], theta_1[3],
omega_1[2], omega_1[3]);
109 Vp_2 = fourbar.couplerPointVel(theta_2[2], theta_2[3],
omega_2[2], omega_2[3]);
110 // Store Vp_1 and Vp_2 into the array
111 V_1[i] = cabs(Vp_1);
112 V_2[i] = cabs(Vp_2);
113 }
114 plotV.title("Coupler Point Velocity");
115 plotV.data2D(t, V_1);
116 plotV.data2D(t, V_2);
117 plotV.legend("Solution 1", 0);
118 plotV.legend("Solution 2", 1);
119 plotV.label(PLOT_AXIS_X, "Time [s]");
120 plotV.label(PLOT_AXIS_Y, "Velocity [m/s]");
121 plotV.plotting();
122
123 return 0;
124 }

```

```

1 Solution 1:
2 theta [rad] =      0.2618      0.7854      1.5147      2.3769
3 Solution 2:
4 theta [rad] =      0.2618      0.7854     -1.5850     -2.4472
5
6 Solution 1:
7 P = (    0.0029,    0.0365) [m]
8 Solution 2:
9 P = (    0.0263,   -0.0077) [m]
10
11 Solution 1:
12 gamma =    0.8622 [rad] =   49.4011 [deg]
13 Solution 2:
14 gamma =    0.8622 [rad] =   49.4011 [deg]
15
16 Solution 1:
17 omega [rad/s] =    0.0000   15.0000   -13.1675   -5.8506
18 Vp = (    0.0817,    0.3605) [m/s] =    0.3696 [m/s]
19 Solution 2:
20 omega [rad/s] =    0.0000   15.0000    1.1971   -6.1197
21 Vp = (   -0.1860,    0.2267) [m/s] =    0.2933 [m/s]
22
23 Solution 1:
24 alpha [rad/s^2] =    0.0000    0.0000   76.8866  221.0994
25 Ap = (   -2.9445,   -7.9177) [m/s^2] =    8.4475 [m/s^2]
26 Solution 2:
27 alpha [rad/s^2] =    0.0000    0.0000  269.4943  125.2815
28 Ap = (    2.6825,    0.1347) [m/s^2] =    2.6859 [m/s^2]

```

Problem 7

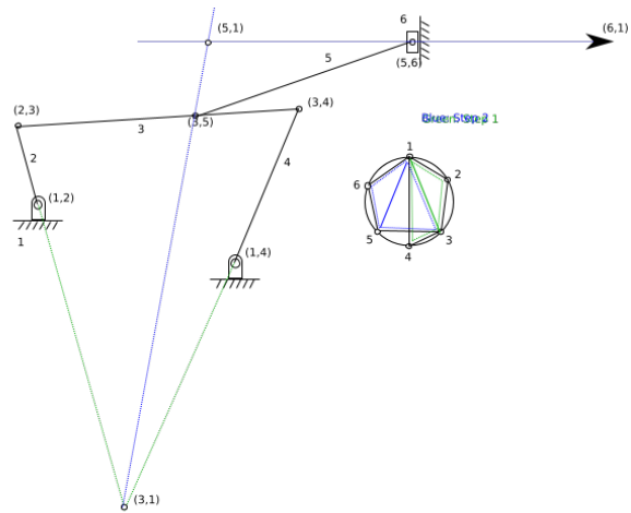


Image 1: Instant center analysis for problem 7.