

HW 7 Solutions

Problem 4

```
1  /**
2   * File: prob4.ch
3   * Author: Nicolas Ventura
4   * Calculate the unknowns of
5   * a fourbar linkage.
6   */
7  #include <stdio.h>
8  #include <fourbar.h>
9  #include <math.h>
10
11 int main(void) {
12     CFourbar fourbar;
13     double theta_1[1:4], theta_2[1:4], r[1:4], rp, beta, gamma_1,
14         gamma_2;
15     double complex P_1, P_2;
16
17     /* Define link lengths */
18     r[1] = 5.0;
19     r[2] = 2.0;
20     r[3] = 3.0;
21     r[4] = 4.5;
22
23     /* Set theta1 and theta2 for both solutions */
24     theta_1[1] = deg2rad(15.0);
25     theta_2[1] = deg2rad(15.0);
26     theta_1[2] = deg2rad(45.0);
27     theta_2[2] = deg2rad(45.0);
28
29     /* Set coupler point */
30     rp = 2.5;
31     beta = deg2rad(30.0);
32
33     /* Set up the fourbar */
34     fourbar.uscUnit(false);
35     fourbar.setLinks(r[1], r[2], r[3], r[4], theta_1[1]);
36     fourbar.setCouplerPoint(rp, beta);
37     /* Part (a) */
38     fourbar.angularPos(theta_1, theta_2, FOURBAR_LINK2);
39     /* Part (b) */
40     fourbar.couplerPointPos(theta_1[2], P_1, P_2);
41     /* Part (c) */
42     fourbar.transAngle(gamma_1, gamma_2, theta_1[2], FOURBAR_LINK2);
43
44     /* Display solutions */
45     printf("\nFirst Solution:\n");
46     printf("Theta3 = %lf rad = %lf deg\n", theta_1[3], rad2deg(
47         theta_1[3]));
48     printf("Theta4 = %lf rad = %lf deg\n", theta_1[4], rad2deg(
49         theta_1[4]));
50     printf("P = (%lf cm, %lf cm)\n", real(P_1), imag(P_1));
51     printf("Gamma = %lf rad = %lf deg\n", gamma_1, rad2deg(gamma_1)
52 );
53 }
```

```

49
50     printf("\nSecond Solution:\n");
51     printf("Theta3 = %lf rad = %lf deg\n", theta_2[3], rad2deg(
        theta_2[3]));
52     printf("Theta4 = %lf rad = %lf deg\n", theta_2[4], rad2deg(
        theta_2[4]));
53     printf("P = (%lf cm, %lf cm)\n", real(P_2), imag(P_2));
54     printf("Gamma = %lf rad = %lf deg\n", gamma_2, rad2deg(gamma_2)
        );
55
56     return 0;
57 }

```

```

1 First Solution:
2 Theta3 = 1.514694 rad = 86.785589 deg
3 Theta4 = 2.376906 rad = 136.186695 deg
4 P = (0.287581 cm, 3.645962 cm)
5 Gamma = 0.862212 rad = 49.401106 deg
6
7 Second Solution:
8 Theta3 = -1.585004 rad = -90.814052 deg
9 Theta4 = -2.447216 rad = -140.215158 deg
10 P = (2.633327 cm, -0.768391 cm)
11 Gamma = 0.862212 rad = 49.401106 deg

```

Problem 7

```

1
2 /**
3  * File: prob7.ch
4  * Animate a quickreturn mechanism
5  * with theta2=45 and in steps of
6  * 1 degrees.
7  * Author: Nicolas Ventura
8  */
9
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <math.h>
13 #include <complex.h>
14 #include <array.h>
15 #include <chplot.h>
16
17 #define NUMPOINTS 361
18
19 int main(void) {
20     int n1, n2;
21     double r [1:7], theta1, rp, beta, theta6;
22     complex double L [1:8];
23
24     int i;
25     double theta2[NUMPOINTS],
26            theta3_1[NUMPOINTS], theta3_2[NUMPOINTS],
27            x1, x2, x3, x4;
28     array double theta4_1[NUMPOINTS], theta4_2[NUMPOINTS],
29            theta5_1_1[NUMPOINTS], theta5_1_2[NUMPOINTS],
30            theta5_2_1[NUMPOINTS], theta5_2_2[NUMPOINTS];

```

```

31 double complex z,
32     P_1[NUMPOINTS], P_2[NUMPOINTS],
33     B_1[NUMPOINTS], B_2[NUMPOINTS];
34
35 array double r6_1_1[NUMPOINTS], r6_1_2[NUMPOINTS],
36     r6_2_1[NUMPOINTS], r6_2_2[NUMPOINTS];
37 class CPlot mainplot1;
38 class CPlot mainplot2;
39 class CPlot *subplot;
40 int branches[NUMPOINTS];
41
42 FILE *fp;
43 double sliderWidth, sliderHeight;
44
45 /*
46  * Determine theta4, point B on the output link, and coupler
47  * point P *
48  */
49 // Define the fourbar linkage
50 n1 = 2;
51 n2 = 4;
52 r[1] = 6;
53 r[2] = 1;
54 r[3] = 3;
55 r[4] = 5;
56 r[5] = 4;
57 r[7] = r[4] - 1; /* Length of BO - C */
58 theta1 = deg2rad(-30);
59 rp = 2.5;
60 beta = deg2rad(20);
61 theta6 = 0;
62
63 // Analyze the fourbar with theta2 rotating from 0-360
64 for (i = 0; i <= 360; i++) {
65     theta2[i] = deg2rad(i);
66
67     // Determine theta3 and theta4
68     z = polar(r[1], theta1) - polar(r[2], theta2[i]);
69     branches[i] = complexsolve(n1, n2, r[3], -r[4], z, x1, x2,
70 x3, x4);
71
72     // First set of solutions
73     theta3_1[i] = x1;
74     theta4_1[i] = x2;
75     P_1[i] = polar(r[2], theta2[i]) + polar(rp, theta3_1[i] +
76 beta);
77     B_1[i] = polar(r[2], theta2[i]) + polar(r[3], theta3_1[i]);
78
79     // Second set of solutions
80     theta3_2[i] = x3;
81     theta4_2[i] = x4;
82     P_2[i] = polar(r[2], theta2[i]) + polar(rp, theta3_2[i] +

```

```

beta);
81     B_2[i] = polar(r[2], theta2[i]) + polar(r[3], theta3_2[i]);
82
83     // Find solutions for r5 and r6 : Circuit 1
84     complexsolve(1, 4, theta6, -r[5],
85                 polar(r[1], theta1) + polar(r[7], theta4_1[i])
86
87                 ,
88                 x1, x2, x3, x4);
89     r6_1_1[i] = x1;
90     theta5_1_1[i] = x2;
91     r6_1_2[i] = x3;
92     theta5_1_2[i] = x4;
93
94     // Find solutions for r5 and r6 : Circuit 2
95     complexsolve(1, 4, theta6, -r[5],
96                 polar(r[1], theta1) + polar(r[7], theta4_2[i])
97
98                 ,
99                 x1, x2, x3, x4);
100    r6_2_1[i] = x1;
101    theta5_2_1[i] = x2;
102    r6_2_2[i] = x3;
103    theta5_2_2[i] = x4;
104
105    // Smooth theta4_1, theta4_2
106    unwrap(theta4_1, theta4_1);
107    unwrap(theta4_2, theta4_2);
108    unwrap(theta5_1_2, theta5_1_2);
109
110    /* Part (a) */
111
112    printf("At 45 degrees, \n");
113    printf("branches = %d\n", branches[45]);
114    printf("Circuit 1:\n");
115    printf("theta3 = %lf\n", theta3_1[45]);
116    printf("theta4 = %lf\n", theta4_1[45]);
117    printf("theta5 = %lf\n", theta5_1_1[45]);
118    printf("r6 = %lf\n", r6_1_1[45]);
119    printf("P = %lf\n", P_1[45]);
120    printf("\nCircuit 2:\n");
121    printf("theta3 = %lf\n", theta3_2[45]);
122    printf("theta4 = %lf\n", theta4_2[45]);
123    printf("theta5 = %lf\n", theta5_2_1[45]);
124    printf("r6 = %lf\n", r6_2_1[45]);
125    printf("P = %lf\n", P_2[45]);
126    printf("\nCircuit 3:\n");
127    printf("theta3 = %lf\n", theta3_1[45]);
128    printf("theta4 = %lf\n", theta4_1[45]);
129    printf("theta5 = %lf\n", theta5_1_2[45]);
130    printf("r6 = %lf\n", r6_1_2[45]);
131    printf("P = %lf\n", P_1[45]);
132    printf("\nCircuit 4:\n");
133    printf("theta3 = %lf\n", theta3_2[45]);
134    printf("theta4 = %lf\n", theta4_2[45]);
135    printf("theta5 = %lf\n", theta5_2_2[45]);
136    printf("r6 = %lf\n", r6_2_2[45]);
137    printf("P = %lf\n", P_2[45]);

```

```

135  /* Part (b) */
136
137  fp = fopen("partb.qnm", "w");
138  if (fp == NULL) {
139      printf("Could not open animation file.\n");
140      exit(1);
141  }
142
143  sliderWidth = 0.30;
144  sliderHeight = 0.20;
145
146  L[1] = complex(0, 0);
147  L[2] = polar(r[2], deg2rad(45.0));
148  L[3] = L[2] + polar(r[3], theta3_1[45]);
149  L[4] = L[3] - polar(r[4], theta4_1[45]);
150  L[5] = L[4] + polar(r[7], theta4_1[45]); // r4'
151  L[6] = complex(r6_1_1[45], 0);
152  L[7] = P_1[45];
153
154  fprintf(fp, "title \"Part B\"\n");
155  fprintf(fp, "fixture\n");
156  fprintf(fp, "groundpin 0 0\n");
157  fprintf(fp, "groundpin %lf %lf\n", real(L[4]), imag(L[4]));
158  fprintf(fp, "ground %lf %lf %lf %lf\n",
159      real(L[6]) - sliderWidth, imag(L[6]) - sliderHeight /
160      2, real(L[6]) + sliderWidth, imag(L[6]) - sliderHeight / 2);
161  fprintf(fp, "link 0 0 %lf %lf %lf %lf %lf %lf\n",
162      real(L[2]), imag(L[2]),
163      real(L[3]), imag(L[3]),
164      real(L[4]), imag(L[4]));
165  fprintf(fp, "link %lf %lf %lf %lf\n",
166      real(L[5]), imag(L[5]),
167      real(L[6]), imag(L[6]));
168  fprintf(fp, "rectangle %lf %lf %lf %lf\n",
169      real(L[6]) - sliderWidth / 2, imag(L[6]) - sliderHeight
170      / 2, sliderWidth, sliderHeight);
171  fprintf(fp, "polygon fill gray90 %lf %lf %lf %lf %lf %lf\n",
172      real(L[2]), imag(L[2]),
173      real(L[3]), imag(L[3]),
174      real(L[7]), imag(L[7]));
175  fprintf(fp, "point %lf %lf\n", real(L[7]), imag(L[7]));
176
177  fclose(fp);
178
179  qanimate partb.qnm
180
181  /* Part (c) */
182
183  fp = fopen("partc.qnm", "w");
184  if (fp == NULL) {
185      printf("Could not open animation file.\n");
186      exit(1);
187  }
188
189  sliderWidth = 0.30;
190  sliderHeight = 0.20;

```

```

190 fprintf(fp, "title \"Part C\\n\\n");
191 fprintf(fp, "fixture\\n");
192 fprintf(fp, "groundpin 0 0\\n");
193 fprintf(fp, "groundpin %lf %lf\\n", real(L[4]), imag(L[4]));
194 fprintf(fp, "ground %lf %lf %lf %lf\\n",
195         min(r6_1_1) - sliderWidth / 2, -sliderHeight / 2, max(
r6_1_1) + sliderWidth / 2, -sliderHeight / 2);
196 fprintf(fp, "animate restart\\n\\n");
197
198 for (i = 0; i < NUMPOINTS; i++) {
199     fprintf(fp, "# frame %d\\n", i);
200
201     L[1] = complex(0, 0);
202     L[2] = polar(r[2], deg2rad((double)(i)));
203     L[3] = L[2] + polar(r[3], theta3_1[i]);
204     L[4] = L[3] - polar(r[4], theta4_1[i]);
205     L[5] = L[4] + polar(r[7], theta4_1[i]); // r4'
206     L[6] = complex(r6_1_1[i], 0);
207     L[7] = P_1[i];
208
209     fprintf(fp, "link 0 0 %lf %lf %lf %lf %lf %lf\\n",
210             real(L[2]), imag(L[2]),
211             real(L[3]), imag(L[3]),
212             real(L[4]), imag(L[4]));
213     fprintf(fp, "link %lf %lf %lf %lf\\n",
214             real(L[5]), imag(L[5]),
215             real(L[6]), imag(L[6]));
216     fprintf(fp, "rectangle %lf %lf %lf %lf\\n",
217             real(L[6]) - sliderWidth / 2, imag(L[6]) -
sliderHeight / 2, sliderWidth, sliderHeight);
218     fprintf(fp, "polygon fill gray90 %lf %lf %lf %lf %lf %lf\\n",
219             real(L[2]), imag(L[2]),
220             real(L[3]), imag(L[3]),
221             real(L[7]), imag(L[7]));
222     fprintf(fp, "point trace %lf %lf\\n", real(L[7]), imag(L
[7]));
223     fprintf(fp, "\\n");
224 }
225
226 fclose(fp);
227
228 qanimate partc.qnm
229
230 return 0;
231 }

```

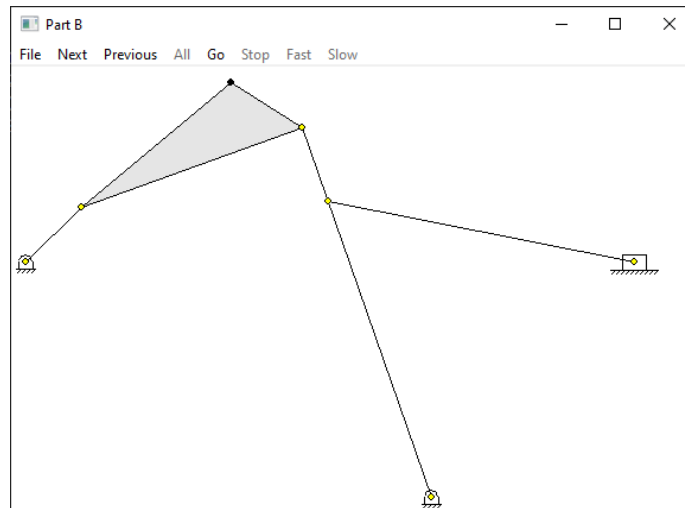


Image 1: Problem 7(b)

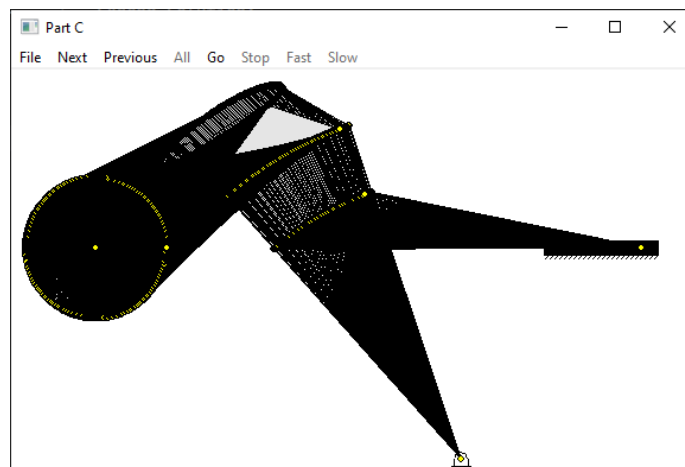


Image 2: Problem 7(c)