HW 7 Solutions

Problem 4

```
/**
 * File: prob4.ch
 * Author: Nicolas Ventura
 * Calculate the unknowns of
 * a fourbar linkage.
 */
#include <stdio.h>
#include <fourbar.h>
#include <math.h>

int main(void) {
    CFourbar fourbar;
    double theta_1[1:4], theta_2[1:4], r[1:4], rp, beta, gamma_1,
    gamma_2;
    double complex P_1, P_2;

    /* Define link lengths */
    r[1] = 5.0;
    r[2] = 2.0;
    r[3] = 3.0;
    r[4] = 4.5;

    /* Set theta1 and theta2 for both solutions */
    theta_1[1] = deg2rad(15.0);
    theta_2[1] = deg2rad(15.0);
    theta_1[2] = deg2rad(45.0);
    theta_2[2] = deg2rad(45.0);

    /* Set coupler point */
    rp = 2.5;
    beta = deg2rad(30.0);

    /* Set up the fourbar */
    fourbar.uscUnit(false);
    fourbar.setLinks(r[1], r[2], r[3], r[4], theta_1[1]);
    fourbar.setCouplerPoint(rp, beta);
    /* Part (a) */
    fourbar.angularPos(theta_1, theta_2, FOURBAR_LINK2);
    /* Part (b) */
    fourbar.couplerPointPos(theta_1[2], P_1, P_2);
    /* Part (c) */
    fourbar.transAngle(gamma_1, gamma_2, theta_1[2], FOURBAR_LINK2)
    ;

    /* Display solutions */
    printf("\nFirst Solution:\n");
    printf("Theta3 = %lf rad = %lf deg\n", theta_1[3], rad2deg(
    theta_1[3]));
    printf("Theta4 = %lf rad = %lf deg\n", theta_1[4], rad2deg(
    theta_1[4]));
    printf("P = (%lf cm, %lf cm)\n", real(P_1), imag(P_1));
    printf("Gamma = %lf rad = %lf deg\n", gamma_1, rad2deg(gamma_1)
    );
```

```

    printf("\nSecond Solution:\n");
    printf("Theta3 = %lf rad = %lf deg\n", theta_2[3], rad2deg(
    theta_2[3]));
    printf("Theta4 = %lf rad = %lf deg\n", theta_2[4], rad2deg(
    theta_2[4]));
    printf("P = (%lf cm, %lf cm)\n", real(P_2), imag(P_2));
    printf("Gamma = %lf rad = %lf deg\n", gamma_2, rad2deg(gamma_2)
    );

    return 0;
}
```

```
First Solution:
Theta3 = 1.514694 rad = 86.785589 deg
Theta4 = 2.376906 rad = 136.186695 deg
P = (0.287581 cm, 3.645962 cm)
Gamma = 0.862212 rad = 49.401106 deg

Second Solution:
Theta3 = -1.585004 rad = -90.814052 deg
Theta4 = -2.447216 rad = -140.215158 deg
P = (2.633327 cm, -0.768391 cm)
Gamma = 0.862212 rad = 49.401106 deg
```

Problem 7

```

/**
 * File: prob7.ch
 * Animate a quickreturn mechanism
 * with theta2=45 and in steps of
 * 1 degrees.
 * Author: Nicolas Ventura
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>
#include <array.h>
#include <chplot.h>

#define NUMPOINTS 361

int main(void) {
    int n1, n2;
    double r [1:7], theta1, rp, beta, theta6;
    complex double L [1:8];

    int i;
    double theta2[NUMPOINTS],
        theta3_1[NUMPOINTS], theta3_2[NUMPOINTS],
        x1, x2, x3, x4;
    array double theta4_1[NUMPOINTS], theta4_2[NUMPOINTS],
        theta5_1_1[NUMPOINTS], theta5_1_2[NUMPOINTS],
        theta5_2_1[NUMPOINTS], theta5_2_2[NUMPOINTS];
```

```
    double complex z,
        P_1[NUMPOINTS], P_2[NUMPOINTS],
        B_1[NUMPOINTS], B_2[NUMPOINTS];

    array double r6_1_1[NUMPOINTS], r6_1_2[NUMPOINTS],
        r6_2_1[NUMPOINTS], r6_2_2[NUMPOINTS];
    class CPlot mainplot1;
    class CPlot mainplot2;
    class CPlot *subplot;
    int branches[NUMPOINTS];

    FILE *fp;
    double sliderWidth, sliderHeight;

    /*
    ********************************************************************

     * Determine theta4, point B on the output link, and coupler
    point P *

    ********************************************************************
    */

    // Define the fourbar linkage
    n1 = 2;
    n2 = 4;
    r[1] = 6;
    r[2] = 1;
    r[3] = 3;
    r[4] = 5;
    r[5] = 4;
    r[7] = r[4] - 1; /* Length of B0 - C */
    theta1 = deg2rad(-30);
    rp = 2.5;
    beta = deg2rad(20);
    theta6 = 0;

    // Analyze the fourbar with theta2 rotating from 0-360
    for (i = 0; i <= 360; i++) {
        theta2[i] = deg2rad(i);

        // Determine theta3 and theta4
        z = polar(r[1], theta1) - polar(r[2], theta2[i]);
        branches[i] = complexsolve(n1, n2, r[3], -r[4], z, x1, x2,
    x3, x4);

        // First set of solutions
        theta3_1[i] = x1;
        theta4_1[i] = x2;
        P_1[i] = polar(r[2], theta2[i]) + polar(rp, theta3_1[i] +
    beta);
        B_1[i] = polar(r[2], theta2[i]) + polar(r[3], theta3_1[i]);

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

```
    /* Part (b) */

    fp = fopen("partb.qnm", "w");
    if (fp == NULL) {
        printf("Could not open animation file.\n");
        exit(1);
    }

    sliderWidth = 0.30;
    sliderHeight = 0.20;

    L[1] = complex(0, 0);
    L[2] = polar(r[2], deg2rad(45.0));
    L[3] = L[2] + polar(r[3], theta3_1[45]);
    L[4] = L[3] - polar(r[4], theta4_1[45]);
    L[5] = L[4] + polar(r[7], theta4_1[45]); // r4'
    L[6] = complex(r6_1_1[45], 0);
    L[7] = P_1[45];

    fprintf(fp, "title \"Part B\"\n");
    fprintf(fp, "fixture\n");
    fprintf(fp, "groundpin 0 0\n");
    fprintf(fp, "groundpin %lf %lf\n", real(L[4]), imag(L[4]));
    fprintf(fp, "ground %lf %lf %lf %lf\n",
            real(L[6]) - sliderWidth, imag(L[6]) - sliderHeight /
2, real(L[6]) + sliderWidth, imag(L[6]) - sliderHeight / 2);
    fprintf(fp, "link 0 0 %lf %lf %lf %lf %lf %lf\n",
            real(L[2]), imag(L[2]),
            real(L[3]), imag(L[3]),
            real(L[4]), imag(L[4]));
    fprintf(fp, "link %lf %lf %lf %lf\n",
            real(L[5]), imag(L[5]),
            real(L[6]), imag(L[6]));
    fprintf(fp, "rectangle %lf %lf %lf %lf\n",
            real(L[6]) - sliderWidth / 2, imag(L[6]) - sliderHeight
 / 2, sliderWidth, sliderHeight);
    fprintf(fp, "polygon fill gray90 %lf %lf %lf %lf %lf %lf\n",
            real(L[2]), imag(L[2]),
            real(L[3]), imag(L[3]),
            real(L[7]), imag(L[7]));
    fprintf(fp, "point %lf %lf\n", real(L[7]), imag(L[7]));

    fclose(fp);

    qanimate partb.qnm

    /* Part (c) */

    fp = fopen("partc.qnm", "w");
    if (fp == NULL) {
        printf("Could not open animation file.\n");
        exit(1);
    }

    sliderWidth = 0.30;
    sliderHeight = 0.20;

```

```
    fprintf(fp, "title \"Part C\"\n");
    fprintf(fp, "fixture\n");
    fprintf(fp, "groundpin 0 0\n");
    fprintf(fp, "groundpin %lf %lf\n", real(L[4]), imag(L[4]));
    fprintf(fp, "ground %lf %lf %lf %lf\n",
            min(r6_1_1) - sliderWidth / 2, -sliderHeight / 2, max(r6_1_1) + sliderWidth / 2, -sliderHeight / 2);
    fprintf(fp, "animate restart\n\n");

    for (i = 0; i < NUMPOINTS; i++) {
        fprintf(fp, "# frame %d\n", i);

        L[1] = complex(0, 0);
        L[2] = polar(r[2], deg2rad((double)(i)));
        L[3] = L[2] + polar(r[3], theta3_1[i]);
        L[4] = L[3] - polar(r[4], theta4_1[i]);
        L[5] = L[4] + polar(r[7], theta4_1[i]); // r4'
        L[6] = complex(r6_1_1[i], 0);
        L[7] = P_1[i];

        fprintf(fp, "link 0 0 %lf %lf %lf %lf %lf %lf\\\n",
                real(L[2]), imag(L[2]),
                real(L[3]), imag(L[3]),
                real(L[4]), imag(L[4]));
        fprintf(fp, "link %lf %lf %lf %lf\\\n",
                real(L[5]), imag(L[5]),
                real(L[6]), imag(L[6]));
        fprintf(fp, "rectangle %lf %lf %lf %lf\\\n",
                real(L[6]) - sliderWidth / 2, imag(L[6]) - sliderHeight / 2, sliderWidth, sliderHeight);
        fprintf(fp, "polygon fill gray90 %lf %lf %lf %lf %lf %lf\\\n",
                real(L[2]), imag(L[2]),
                real(L[3]), imag(L[3]),
                real(L[7]), imag(L[7]));
        fprintf(fp, "point trace %lf %lf\\\n", real(L[7]), imag(L[7]));
        fprintf(fp, "\n");
    }

    fclose(fp);

    qanimate partc.qnm

    return 0;
}
```

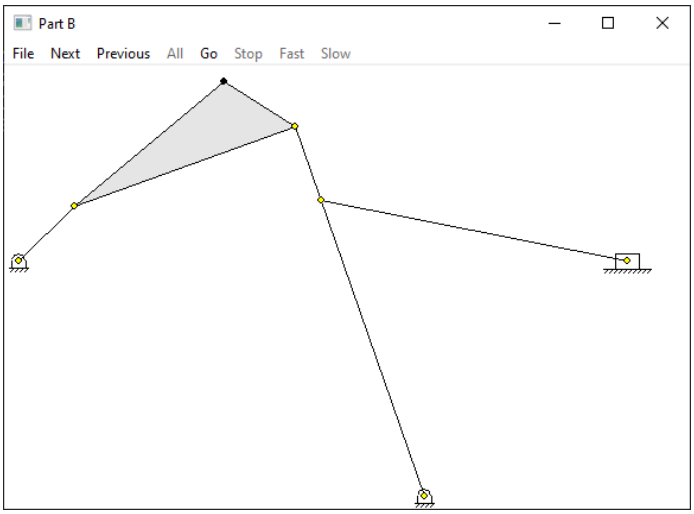

Image 1: Problem 7(b)

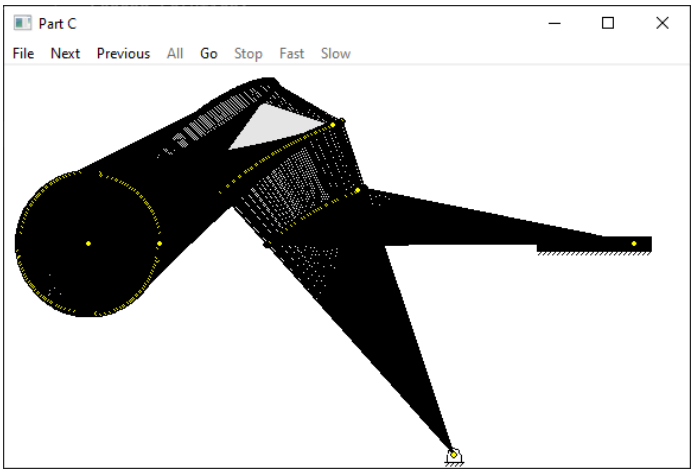

Image 2: Problem 7(c)