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.

```c
/**
 * File: quick.ch
 * Author: Nicolas Ventura
 */

#include "quickreturn.h" // Include the header file

int main(void) {
    CQuickReturn qr;
    int numpoints = 100; // Number of frames to animate

    bool unit = SI; // SI units are in meters
    double r1 = 2.5/100,
        r2 = 1.0/100,
        r4 = 6.5/100,
        r5 = 3.0/100,
        r6,
        r7 = 5.0/100;
    double theta1 = deg2rad(90.0),
        theta2 = deg2rad(30.0),
        theta4, theta5;

    qr.uscUnit(unit);
    qr.setLinks(r1, r2, r4, r5, r7, theta1);
    qr.setNumPoints(numpoints);

    /* Problem 5 */
    theta4 = qr.getAngPos(theta2, QR_LINK_4);
    theta5 = qr.getAngPos(theta2, QR_LINK_5);
    r6 = qr.sliderPos(theta2);
    printf("Theta4 = %lf rad = %lf deg\n", theta4, rad2deg(theta4));
    printf("Theta5 = %lf rad = %lf deg\n", theta5, rad2deg(theta5));
    printf("r6 = %lf m\n", r6);
    /* Part (a) */
    qr.displayPosition(theta2, QANIMATE_OUTPUTTYPE_DISPLAY);
    /* Part (b) */
    qr.animation(QANIMATE_OUTPUTTYPE_DISPLAY);
    return 0;
}
```

```
Theta4 = 1.289761 rad = 73.897886 deg
Theta5 = -0.427942 rad = -24.519274 deg
r6 = 0.045322 m
```

Problem 6

```
/**
 * File: prob6.ch
 * Author: Nicolas Ventura
 * Run a kinematic analysis
 * on a standard crank-rocker
 * fourbar linkage.
 */

#include <fourbar.h>

#define NUMPOINTS 100

int main(void) {
    //// Create variables ////
    CFourbar fourbar;
    CPlot plot1, plot2, plotV;
    int i;
    double r[1:4], rp, beta, theta2[NUMPOINTS], t[NUMPOINTS];
    ////    SOLUTION 1           SOLUTION 2    ////
    array double
            theta_1[1:4],        theta_2[1:4],
            omega_1[1:4],        omega_2[1:4],
            alpha_1[1:4],        alpha_2[1:4],
            V_1[NUMPOINTS],      V_2[NUMPOINTS],
            theta3[NUMPOINTS],   theta4[NUMPOINTS];
    double
            gamma_1,             gamma_2;
    complex double
            P_1,                 P_2,
            Vp_1,                Vp_2,
            Ap_1,                Ap_2;

    //// Define known values ////
    r[1] = 5.0 / 100;
    r[2] = 2.0 / 100;
    r[3] = 3.0 / 100;
    r[4] = 4.5 / 100;
    rp = 2.5 / 100;
    beta = deg2rad(30.0);
    linspace(theta2, deg2rad(0.0), deg2rad(360.0));

    //// Set knowns for first solution ////
    theta_1[1] = deg2rad(15.0);
    theta_1[2] = deg2rad(45.0);
    omega_1[1] = 0.0; // rad/sec
    omega_1[2] = 15.0; // rad/sec
    alpha_1[1] = 0.0; // rad/s^2
    alpha_1[2] = 0.0; // rad/s^2

    //// Set knowns for second solution ////
    theta_2[1] = deg2rad(15.0);
    theta_2[2] = deg2rad(45.0);
    omega_2[1] = 0.0; // rad/sec
    omega_2[2] = 15.0; // rad/sec
    alpha_2[1] = 0.0; // rad/s^2
    alpha_2[2] = 0.0; // rad/s^2
```

```
    //// Basic fourbar setup ////
    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);
    printf("Solution 1:\n  theta [rad] = %9.4lf", theta_1);
    printf("Solution 2:\n  theta [rad] = %9.4lf\n", theta_2);

    //// Part (b) ////
    fourbar.couplerPointPos(theta_1[2], P_1, P_2);
    printf("Solution 1:\n  P = (%9.4lf, %9.4lf) [m]\n", real(P_1), imag(P_1));
    printf("Solution 2:\n  P = (%9.4lf, %9.4lf) [m]\n\n", real(P_2), imag(P_2));

    //// Part (c) ////
    fourbar.transAngle(gamma_1, gamma_2, theta_1[2], FOURBAR_LINK2);
    printf("Solution 1:\n  gamma = %9.4lf [rad] = %9.4lf [deg]\n", gamma_1, rad2deg(gamma_1));
    printf("Solution 2:\n  gamma = %9.4lf [rad] = %9.4lf [deg]\n\n", gamma_2, rad2deg(gamma_2));

    //// Part (d) ////
    fourbar.angularVel(theta_1, omega_1, FOURBAR_LINK2);
    fourbar.angularVel(theta_2, omega_2, FOURBAR_LINK2);
    Vp_1 = fourbar.couplerPointVel(theta_1[2], theta_1[3], omega_1[2], omega_1[3]);
    Vp_2 = fourbar.couplerPointVel(theta_2[2], theta_2[3], omega_2[2], omega_2[3]);
    printf("Solution 1:\n  omega [rad/s] = %9.4lf  Vp = (%9.4lf, %9.4lf) [m/s] = %9.4lf [m/s]\n", omega_1, real(Vp_1), imag(Vp_1), cabs(Vp_1));
    printf("Solution 2:\n  omega [rad/s] = %9.4lf  Vp = (%9.4lf, %9.4lf) [m/s] = %9.4lf [m/s]\n\n", omega_2, real(Vp_2), imag(Vp_2), cabs(Vp_2));

    //// Part (e) ////
    fourbar.angularAccel(theta_1, omega_1, alpha_1, FOURBAR_LINK2);
    fourbar.angularAccel(theta_2, omega_2, alpha_2, FOURBAR_LINK2);
    Ap_1 = fourbar.couplerPointAccel(theta_1[2], theta_1[3], omega_1[2], omega_1[3], alpha_1[2], alpha_1[3]);
    Ap_2 = fourbar.couplerPointAccel(theta_2[2], theta_2[3], omega_2[2], omega_2[3], alpha_2[2], alpha_2[3]);
    printf("Solution 1:\n  alpha [rad/s^2] = %9.4lf  Ap = (%9.4lf, %9.4lf) [m/s^2] = %9.4lf [m/s^2]\n", alpha_1, real(Ap_1), imag(Ap_1), cabs(Ap_1));
    printf("Solution 2:\n  alpha [rad/s^2] = %9.4lf  Ap = (%9.4lf, %9.4lf) [m/s^2] = %9.4lf [m/s^2]\n", alpha_2, real(Ap_2), imag(Ap_2), cabs(Ap_2));

    //// Part (f) ////
    fourbar.setAngularVel(omega_1[2]);
  fourbar.plotAngularVels(&plot1, 1);
```

```
  fourbar.plotAngularVels(&plot2, 2);
    // Calculate the t array
    fourbar.angularPoss(1, t, theta3, theta4);
    for (i = 0; i < NUMPOINTS; i++) {
        // Need to re-calculate for every theta2
        theta_1[2] = theta2[i];
        theta_2[2] = theta2[i];
        fourbar.angularPos(theta_1, theta_2, FOURBAR_LINK2);
        fourbar.angularVel(theta_1, omega_1, FOURBAR_LINK2);
        fourbar.angularVel(theta_2, omega_2, FOURBAR_LINK2);
        Vp_1 = fourbar.couplerPointVel(theta_1[2], theta_1[3],
    omega_1[2], omega_1[3]);
        Vp_2 = fourbar.couplerPointVel(theta_2[2], theta_2[3],
    omega_2[2], omega_2[3]);
        // Store Vp_1 and Vp_2 into the array
        V_1[i] = cabs(Vp_1);
        V_2[i] = cabs(Vp_2);
    }
    plotV.title("Coupler Point Velocity");
    plotV.data2D(t, V_1);
    plotV.data2D(t, V_2);
    plotV.legend("Solution 1", 0);
    plotV.legend("Solution 2", 1);
  plotV.label(PLOT_AXIS_X, "Time [s]");
  plotV.label(PLOT_AXIS_Y, "Velocity [m/s]");
    plotV.plotting();

    return 0;
}
```

```
Solution 1:
  theta [rad] =    0.2618    0.7854    1.5147    2.3769
Solution 2:
  theta [rad] =    0.2618    0.7854   -1.5850   -2.4472

Solution 1:
  P = (   0.0029,    0.0365) [m]
Solution 2:
  P = (   0.0263,   -0.0077) [m]

Solution 1:
  gamma =    0.8622 [rad] =   49.4011 [deg]
Solution 2:
  gamma =    0.8622 [rad] =   49.4011 [deg]

Solution 1:
  omega [rad/s] =    0.0000   15.0000  -13.1675   -5.8506
  Vp = (   0.0817,    0.3605) [m/s] =    0.3696 [m/s]
Solution 2:
  omega [rad/s] =    0.0000   15.0000    1.1971   -6.1197
  Vp = (  -0.1860,    0.2267) [m/s] =    0.2933 [m/s]

Solution 1:
  alpha [rad/s^2] =    0.0000    0.0000   76.8866  221.0994
  Ap = (  -2.9445,   -7.9177) [m/s^2] =    8.4475 [m/s^2]
Solution 2:
  alpha [rad/s^2] =    0.0000    0.0000  269.4943  125.2815
  Ap = (   2.6825,    0.1347) [m/s^2] =    2.6859 [m/s^2]
```

Problem 7

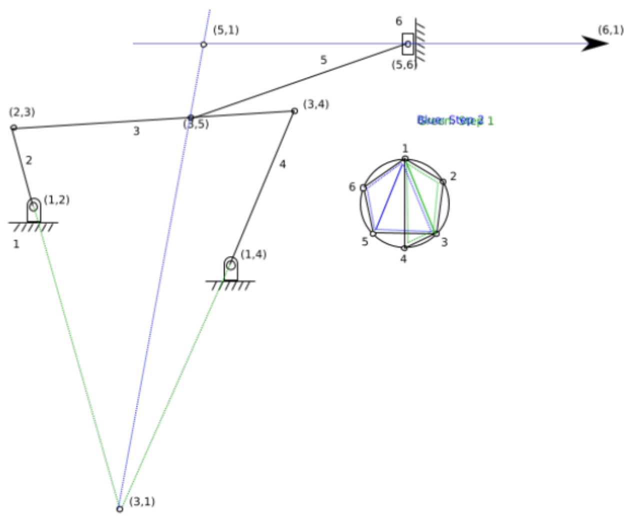

Image 1: Instant center analysis for problem 7.