AEP 4380 HW#4: Ordinary Differential Equations and Chaotic Systems

Gianfranco Grillo

October 5, 2016

1 Background

1.1 Numerical Solving of ODEs Using 4th Order Runge-Kutta

Ordinary differential equations (ODEs) are an essential part of the language used in science and engineering. Although many of these ODEs can be solved analytically, a significant portion of them have solutions that cannot be expressed in terms of closed, elementary functions. Nevertheless, these solutions can be approximated via numerical algorithms. The Runge-Kutta algorithms rank among the most popular of these. In particular, the 4th order Runge-Kutta method has become one of the most trusted numerical algorithms used to solve ODEs and systems of ODEs. The following is an outline of how it works.

Given an ODE of the form

$$\frac{dx}{dt} = f(t,\, x(t)) \tag{1}$$

With initial conditions

$$x(t_0) = x_0 \tag{2}$$

Where t is an independent variable and x is a function of t, the 4th order Runge-Kutta algorithm reconstructs $x(t)$ by slowly incrementing t in several small steps, and attempting to adjust for the change in $\frac{dx}{dt}$ as it changes as a function of t. In order to do this, it relies on four parameters, namely k_0, k_1, k_2, and k_3, which are given by

$$\begin{aligned} k_0 &= hf(t_0, x_0) && (3)\\ k_1 &= hf(t_0 + \tfrac{1}{2}h,\, x_0 + \tfrac{1}{2}k_0) && (4)\\ k_2 &= hf(t_0 + \tfrac{1}{2}h,\, x_0 + \tfrac{1}{2}k_1) && (5)\\ k_3 &= hf(t_0 + h,\, x_0 + k_2) && (6) \end{aligned}$$

Where h corresponds to the step size. These parameters are combined in order to produce the next value of x, x_1, in the following way:

$$x_1 = x(t_0 + h) = x_0 + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3) \tag{7}$$

In general, we have

$$x_{n+1} = x(t_n + h) = x_n + \frac{1}{6}(k_{0n} + 2k_{1n} + 2k_{2n} + k_{3n}) \tag{8}$$

Where k_{in} are the corresponding values of the k parameters as defined in (3)-(6), (i.e. the first one is given by $k_{0n} = hf(t_n, x_n)$). Repeating this multiple times with a sufficiently small step size h in principle allows us to reproduce the form of the function $x(t)$ for a certain range of values of t.

Since higher order differential equations can be expressed as a system of coupled first order differential equations, it is possible to use the same procedure to solve higher order differential equations. Thus, for an N^{th}order differential equation with N initial conditions in the form

$$\frac{d\vec{x}}{dt} = \vec{f}(t, \vec{x}(t)) \tag{9}$$
$$\vec{x}(t_0) = \vec{x_0} \tag{10}$$

We can apply the Runge-Kutta method in order to solve one by one each of the equations in the array. The goal of this homework is to use 4th order Runge-Kutta to numerically solve the third order differential equation of the chaotic Chen-Lee system. Along the way, the algorithm will also be used to solve the second order differential equation describing a harmonic oscillator, as a means to check the correctness of Runge-Kutta's code implementation.

1.2 Undamped 1D Harmonic Oscillator

The differential equation governing the motion of a 1D, undamped harmonic oscillator is

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \tag{11}$$

With initial conditions $y_0 = 1$ and $y_0' = 0$, the system can be solved analytically, and the solution is given by

$$y(t) = \cos(\omega t) \tag{12}$$

Making the substitution $\frac{dx}{dt} = y$ allows us to write (11) as a system of two coupled equations, as follows

$$\frac{d\vec{x}}{dt} = \begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \begin{bmatrix} y \\ -\omega^2 x \end{bmatrix} = \begin{bmatrix} f(t,\, x,\, y) \\ g(t,\, x,\, y) \end{bmatrix} \tag{13}$$

We can apply Runge-Kutta to both elements in the vector equation given in (13), and comparing the numerical solution with the expected analytical solution provides a useful way of checking whether the algorithm is being well implemented.

1.3 Chaotic Chen-Lee System

The Chen-Lee system is governed by a third order ODE, which can be decomposed into the system of three coupled ODEs given by

$$\frac{d\vec{x}}{dt} = \begin{bmatrix} x'(t) \\ y'(t) \\ z'(t) \end{bmatrix} = \begin{bmatrix} ax - yz \\ xz + by \\ \frac{1}{3}xy + cz \end{bmatrix} = \begin{bmatrix} f(t,\, x,\, y,\, z) \\ g(t,\, x,\, y,\, z) \\ j(t,\, x,\, y,\, z) \end{bmatrix} \tag{14}$$

Where a, b, and c are constants. This is a potentially chaotic system for which there is no known analytical solution. The code used for 1.2 will also be used to solve this system.

2 Results

2.1 Task 1

Task 1 involved running hw4a.cpp, which solves the harmonic oscillator problem and produces a file with three columns, one with the values of $x(t)$, another with the values of $x'(t)$, and a third one with the corresponding values of t. I used a stepsize of $h = 0.001$ and an angular frequency of $\omega = 1.0$, and looped over 3000 steps in order to produce values of t in the range $0 < t < 30$. Inspection of the data reveals that the $x(t)$ never exceeds a magnitude

of 1 and, by looking at the values of t at those amplitudes, it can be seen that the period is always extremely close to 2π, which is what we would expect given the nature of the analytical solution. Figure 1 shows a plot of $x(t)$ vs t, and Figure 2 shows a phase space plot of $x'(t)$ vs $x(t)$, which also gives the expected result of a circle that closes on itself.

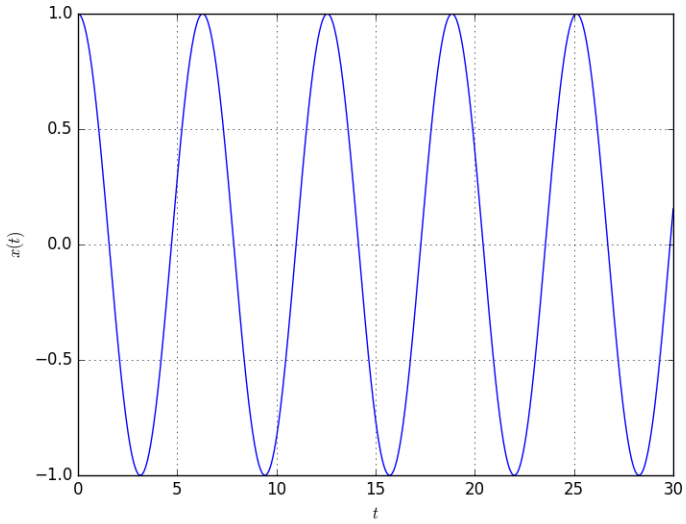

Figure 1: Solution for the harmonic oscillator function as given by (11) using 4th order Runge-Kutta

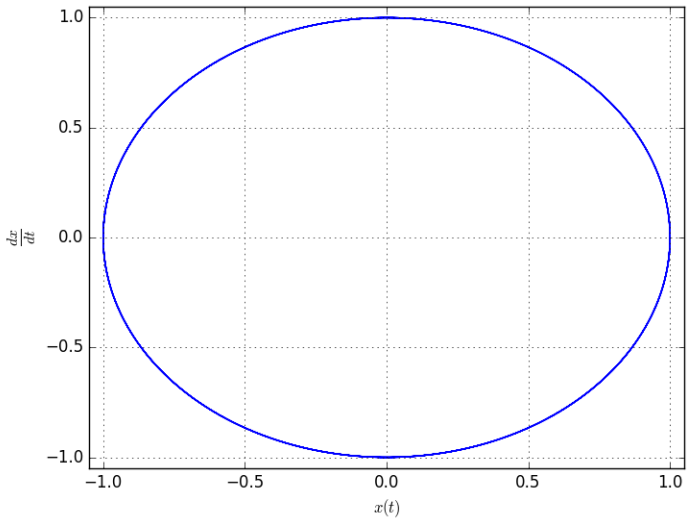

Figure 2: Phase space plot for the harmonic oscillator described in (11) produced using 4th order Runge-Kutta

2.2 Task 2

Running hw4b.cpp produces an output file with four columns, where we can find the numerical values for $x(t)$, $y(t)$, $z(t)$ as a function of t. These were calculated by applying Runge-Kutta to the Chen-Lee system using the parameter

values $(a, b, c) = (3.0, -5.0, -1.0)$ and initial conditions $(x_0, y_0, z_0) = (5, 5, 5)$. I used a step size of $h_1 = 0.001$ in order to plot $x(t)$ vs t in the range $0 < t < 50$, as in Figure 3, and a step size of $h_2 = 0.0005$ in order to make the same plot, as seen in Figure 4.

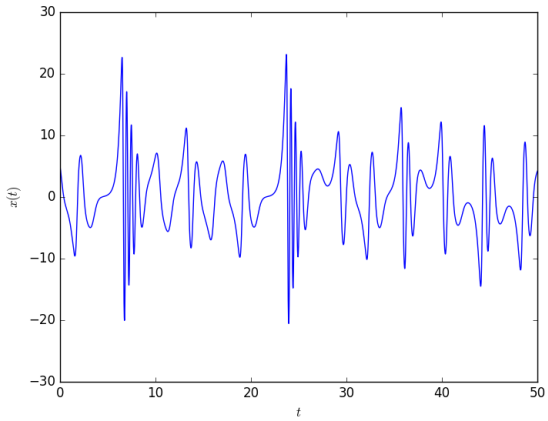

Figure 3: $x(t)$ vs t for Chen-Lee system using step size $h_1 = 0.001$

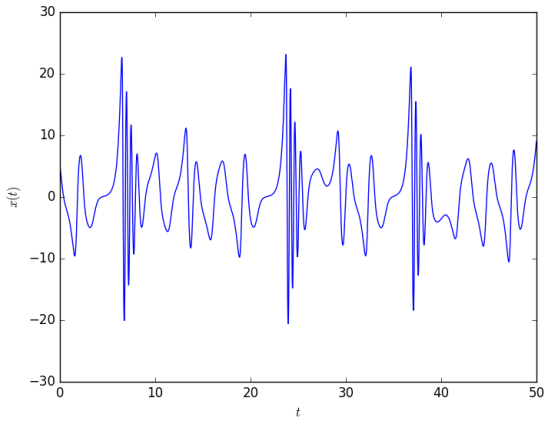

Figure 4: $x(t)$ vs t for Chen-Lee system using step size $h_2 = 0.0005$

Figure 5 shows a plot of $z(t)$ vs t using step size h_1, while Figure 6 shows the same plot with step size h_2. Figure 7 shows a phase space plot of $z(t)$ vs $y(t)$, and Figure 8 shows a phase space plot of $z(t)$ vs $x(t)$, both of them produced using step size h_2. Figure 9 shows a 3D phase space plot of $x(t)$, $y(t)$, and $z(t)$, again using step size h_2.

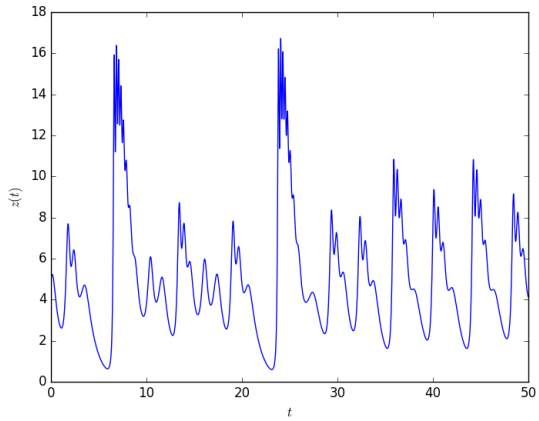

Figure 5: $z(t)$ vs t for chaotic Chen-Lee system using step size $h_1 = 0.001$

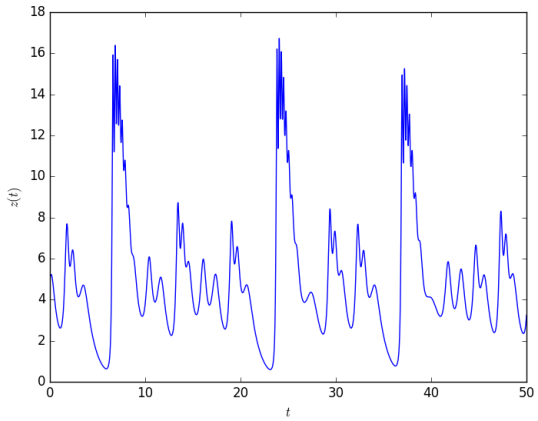

Figure 6: $z(t)$ vs t for chaotic Chen-Lee system using step size $h_2 = 0.0005$

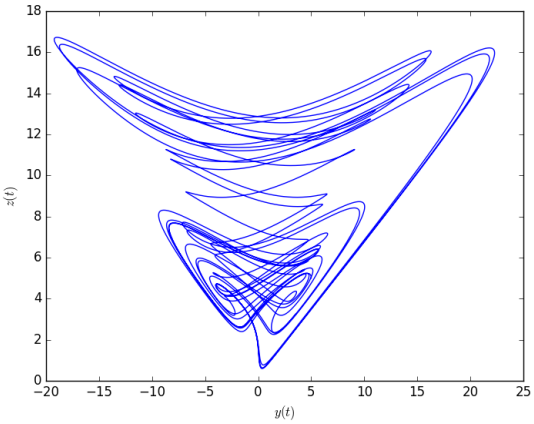

Figure 7: $z(t)$ vs $y(t)$ for chaotic Chen-Lee system

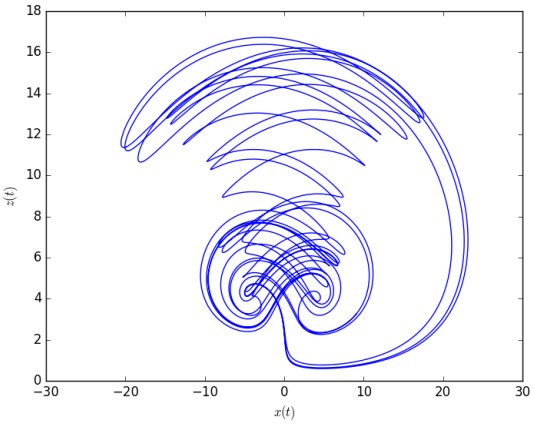

Figure 8: $z(t)$ vs $x(t)$ for chaotic Chen-Lee system

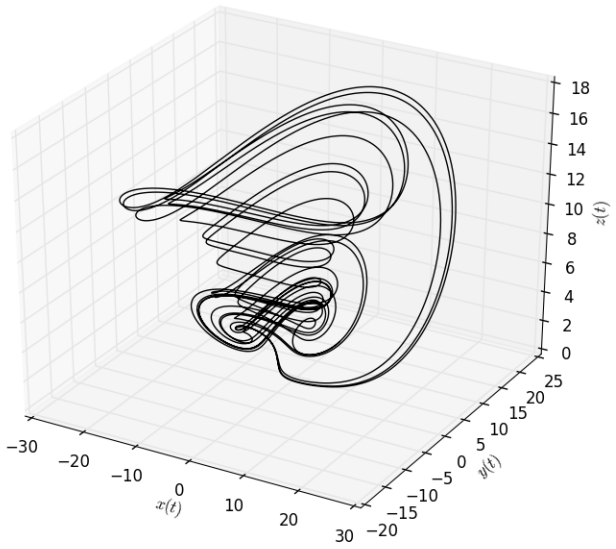

Figure 9: 3D phase space plot for chaotic Chen-Lee system

2.3 Task 3

I tried altering the parameter values (a, b, c) to find a set of values that does not produce a chaotic system like the one above. Setting them equal to (5.0, -10.0, -3.8) accomplishes this objective. This was done just by changing the values of the parameters in the "chenlee" function defined previously in hw4b.cpp. Figures 10 and 11 show the corresponding $z(t)$ vs $y(t)$ and $z(t)$ vs $x(t)$ phase space plots for this system, and Figure 12 shows the corresponding 3D phase space plot. All three were created using a step size equal to h_2. The initial conditions are the same as in

the previous task.

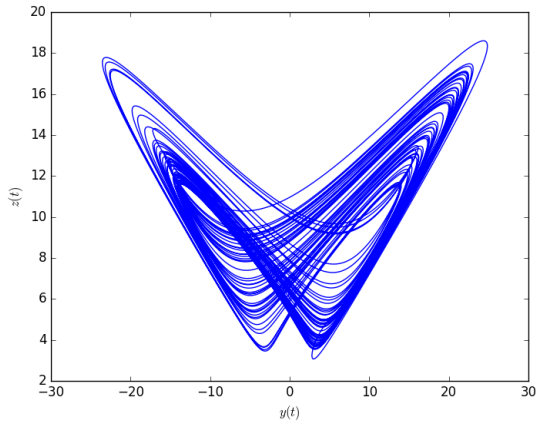

Figure 10: $z(t)$ vs $y(t)$ for non-chaotic C-L system

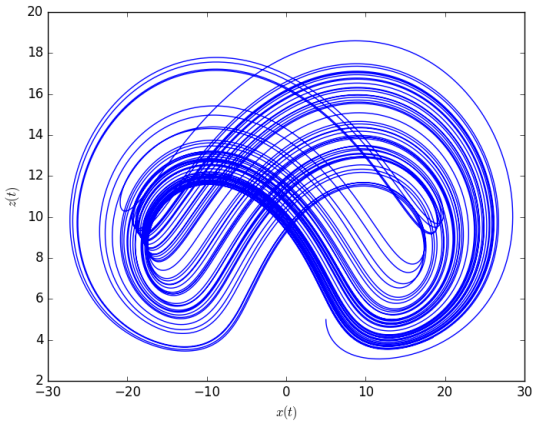

Figure 11: $z(t)$ vs $x(t)$ for non-chaotic C-L system

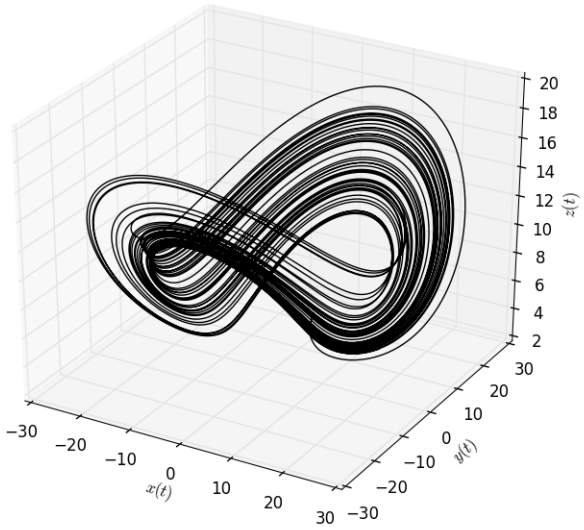

Figure 12: 3D phase space plot for non-chaotic C-L system

3 Analysis

As can be inferred from Figures 3 and 4, the Runge-Kutta numerical solution for the Chen-Lee system with the original parameters is stable only up to $t_1 \approx 35$, as it is only up to that point that both plots appear to be equivalent. After that point, both plots are radically different. I tried a bunch of other values for the h parameter, and in all of them one could observe the same kind of unstable behavior after t_1. This gives us an idea about the limitations

of numerical solutions for ODE equations, especially when the function oscillates wildly, as is the case here. The phase space plots shown in Figures 7, 8, and 9 have the form that we would expect for a chaotic system, as the curve does not close upon itself in the manner that, for instance, a system like the harmonic oscillator shown in Figure 2 does, and instead appears to oscillate wildly around a pair of attractors. Contrast this with the figures produced for Task 3, where the same phase space plots show a much more recognizable pattern. This comparison provides a good example of the striking difference between a chaotic system and a nonchaotic one.

4 Source code

4.1 hw4a.cpp

```
/* AEP 4380 HW#4a
  Runge-Kutta and Chaos
  Run on core i7 with g++ 5.4.0 (Ubuntu)
  Gianfranco Grillo 10/03/2016
*/

#include <cstdlib> // plain C
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output

using namespace std;

int main() {
        int istep, nstep=30000, n = 2, i;
        double initvar[2] = {1.0,0.0}, finalvar[2], t, t0 = 0.0, h=0.001;
        void rk4(double[], double[], int, double, double, void(double[], double,
            double[], double));
        void harmonicf(double[], double, double[], double);
        ofstream fp;
    fp.open( "oscillator.dat" ); // open new file for output
        if( fp.fail() ) { // in case of error
        cout << "cannot open file" << endl;
        return( EXIT_SUCCESS );
        }
        for (istep = 0; istep < nstep; istep++) {
                t = t0 + h*istep;
                rk4(initvar, finalvar, n, h, t, harmonicf);
                for (i = 0; i < n; i++) { initvar[i] = finalvar[i]; }
        fp <<  setw(15) <<  t << setw(15) << finalvar[0] << setw(15) << finalvar[1]
            << setw(15) << endl;
    }
    fp.close();
    return( EXIT_SUCCESS );
}

void rk4(double initvar[], double finalvar[], int n, double h, double t0, void
    functions(double[], double, double[], double)) {
```

```
        int i;
        double *k0, *k1, *k2, *k3, *temp;
        k0 = new double[5*n];
        if (NULL == k0) {
                cout << "can't allocate arrays in rk4" << endl;
                return;
        }
        k1 = k0 + n;
        k2 = k1 + n;
        k3 = k2 + n;
        temp = k3 + n;
        functions(initvar, t0, k0, h);
        for (i = 0; i < n; i++) {
                temp[i] = initvar[i] + 0.5*k0[i];
        }
        functions(temp, t0 + 0.5*h, k1, h);
        for (i = 0; i < n; i++) {
                temp[i] = initvar[i] + 0.5*k1[i];
        }
        functions(temp, t0 + 0.5*h, k2, h);
        for (i = 0; i < n; i++) {
                temp[i] = initvar[i] + k2[i];
        }
        functions(temp, t0 + h, k3, h);
        for (i = 0; i < n; i++) {
                finalvar[i] = initvar[i] + (1.0/6.0)*(k0[i]+2*k1[i]+2*k2[i]+k3[i]);
        }
        delete k0;
        return;
}

void harmonicf(double var[], double t, double k[2], double h) {
        k[0] = h*var[1];
        k[1] = -h*var[0];
        return;
}
```

4.2 hw4b.cpp

```
/* AEP 4380 HW#4b
  Runge-Kutta and Chaos
  Run on core i7 with g++ 5.4.0 (Ubuntu)
  Gianfranco Grillo 10/03/2016
*/

#include <cstdlib> // plain C
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
```

```
using namespace std;

int main() {
        int istep, nstep=100000, n = 3, i;
        double initvar[3] = {5.0,5.0,5.0}, finalvar[3], t, t0 = 0.0, h=0.0005;
        void rk4(double[], double[], int, double, double, void(double[], double,
           double[], double));
        void chenlee(double[], double, double[], double);
        ofstream fp;
    fp.open( "chenlee.dat" ); // open new file for output
        if( fp.fail() ) { // in case of error
        cout << "cannot open file" << endl;
        return( EXIT_SUCCESS );
        }
        for (istep = 0; istep < nstep; istep++) {
                t = t0 + h*istep;
                rk4(initvar, finalvar, n, h, t, chenlee);
                for (i = 0; i < n; i++) { initvar[i] = finalvar[i]; }
        fp <<  setw(15) <<  t << setw(15) << finalvar[0] << setw(15) << finalvar[1]
            << setw(15) << finalvar[2] << endl;
        }
    fp.close();
    return( EXIT_SUCCESS );
}

void rk4(double initvar[], double finalvar[], int n, double h, double t0, void
   functions(double[], double, double[], double)) {
        int i;
        double *k0, *k1, *k2, *k3, *temp;
        k0 = new double[5*n];
        if (NULL == k0) {
                cout << "can't allocate arrays in rk4" << endl;
                return;
        }
        k1 = k0 + n;
        k2 = k1 + n;
        k3 = k2 + n;
        temp = k3 + n;
        functions(initvar, t0, k0, h);
        for (i = 0; i < n; i++) {
                temp[i] = initvar[i] + 0.5*k0[i];
        }
        functions(temp, t0 + 0.5*h, k1, h);
        for (i = 0; i < n; i++) {
                temp[i] = initvar[i] + 0.5*k1[i];
        }
        functions(temp, t0 + 0.5*h, k2, h);
        for (i = 0; i < n; i++) {
                temp[i] = initvar[i] + k2[i];
        }
        functions(temp, t0 + h, k3, h);
        for (i = 0; i < n; i++) {
```

```
                finalvar[i] = initvar[i] + (1.0/6.0)*(k0[i]+2*k1[i]+2*k2[i]+k3[i]);
        }
        delete k0;
        return;
}

void chenlee(double var[], double t, double k[3], double h) {
        double x, y, z;
        x = var[0], y = var[1], z = var[2];
        k[0] = h*(3.0*x-y*z);
        k[1] = h*(x*z-5.0*y);
        k[2] = h*((1.0/3.0)*x*y-z);
        return;
}
```

4.3 hw4c.cpp

Identical to hw4b.cpp but the "chenlee" function is

```
void chenlee(double var[], double t, double k[3], double h) {
        double x, y, z;
        x = var[0], y = var[1], z = var[2];
        k[0] = h*(5.0*x-y*z);
        k[1] = h*(x*z-10.0*y);
        k[2] = h*((1.0/3.0)*x*y-3.8*z);
        return;
}
```

5 References

This homework was completed using material from the lectures and my own understanding of chaotic systems and harmonic oscillators. No bibliographical material was explicitly used.