Homework 2, Computational Physics

Seann Smallwood

October 13, 2020

Abstract

This assignment we were asked to analyze, solve and model three physical systems. The solutions to which all involved differential equations. The three systems were as follows:

1) Double Pendulum
2) Sun, Earth, Satellite
3) Driven Mass-Spring

Lagrangian Mechanics, Newtonian Mechanics and O.D.E solvers in Fortran were utilized to model the systems.

1 Introduction

1.1 Double Pendulum

The Double Pendulum was to be set up in the following manner, defining θ_1 and θ_2 as shown in figure 1.

We had four main objectives for this problem.

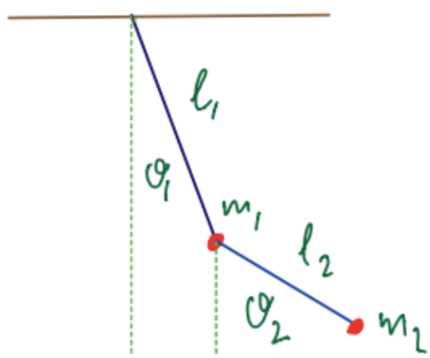


Figure 1: Structure of Double Pendulum system

- Derive the equation for θ_1, θ_2, ω_1, and ω_2
- Derive the expression for total energy
- Write a program to solve the equation of motion for the case where $l_1 = 0.5m$, $l_2 = 0.4m$, $m_1 = 3kg$, $m_2 = 2kg$. Start with initial conditions $\theta_1 = \theta_2 = \pi/10, \pi/6, \pi/4, \pi/2$ and $\omega_1 = \omega_2 = 0$. Follow for 200 seconds.
- When $\theta_1 = 0$ and $\omega_1 > 0$ plot ω_2 as a function of θ_2

1.2 Sun, Earth, Satellite

The objective of this task was to model a system containing the Sun, Earth, and a satellite where $M_{sat} << M_{earth} << M_{sun}$. We were to find the Lagrange points at which the satellite would be in a stationary orbit relative to Earth.

1.3 Driven Mass-Spring

We were tasked with solving and plotting the motion of a spring mass system where the suspension point moves with a constant $w = 7rad/s$, at a radius of $r = 0.5m$.

Given the following parameters and initial conditions, we plotted the motion for 30 seconds. The equilibrium length was $l_0 = 2m$, mass = 1kg, spring constant k = 10m/N, initial angle $\theta_0 = \pi/2$, and the spring was stretched by $\Delta l = 1m$.

1.4 Setup and General Methods

Fortran was used to analyze each system. To implement common mathematical terms and define the precision to which values were calculated a file was created. This file, named numtype, is shown below.

numtype.f90

```fortran
module numtype

    integer,parameter :: dp = selected_real_kind(15,307)
    integer,parameter :: qp = selected_real_kind(33,4931)
    real(dp), parameter :: pi = 4*atan(1._dp)
    complex(dp), parameter :: iic = (0._dp,1._dp)

end module numtype
```

Figure 2: Numtype File

A Makefile, figure 3, was used to compile the fortran code and create and executable file.

The Runge-Kutta method for solving differntial equations was used through a routine we named rk4, figure 4. The functions of which, lines 33 to 43, were

Makefile

```makefile

OBJS1 = numtype.o double_pendulum.o rk4step.o

PROG1 = pend

F90 = gfortran

F90FLAGS = -O3 -funroll-loops  -fexternal-blas

LIBS = -framework Accelerate

LDFLAGS = $(LIBS)

all: $(PROG1)

$(PROG1): $(OBJS1)
	$(F90) $(LDFLAGS) -o $@ $(OBJS1)

clean:
	rm -f $(PROG1) *.{o,mod} fort.*

.SUFFIXES: $(SUFFIXES) .f90

.f90.o:
	$(F90) $(F90FLAGS) -c $<
```

Figure 3: Makefile

changed according to the system being solved. These functions input into rk4 were the equations of motion.

```fortran
subroutine rk4step ( x, h, y )

    use setup
    implicit none
    real(dp), intent(inout) :: x
    real(dp), intent(in) :: h
    real(dp), intent(inout), dimension(n_eq) :: y, f
    real(dp), dimension(n_eq) :: k1, k2, k3, k4, dy

    k1 = kv( x, h, y )
    k2 = kv( x+h/2, h, y + k1/2 )
    k3 = kv( x+h/2, h, y + k2/2 )
    k4 = kv( x+h , h,  y + k3 )
    dy = ( k1 + 2*k2 + 2*k3 + k4 ) / 6

    x = x + h
    y = y + dy

    contains

        function kv(t, dt, y ) result(k)

            use setup
            implicit none
            real(dp), intent(in) :: t, dt
            real(dp), dimension(n_eq), intent(in) :: y
            real(dp), dimension(n_eq) :: f, k

            real(dp) :: r_e, r_e_hat(1:3), r_c, r_c_hat(1:3), r_c_e, r_c_e_hat(1:3)

            !---------------------

            r_e = sqrt( y(1)**2 + y(2)**2 + y(3)**2 )           !magnitude of earths position
            r_e_hat(1:3) = y(1:3) / r_e                         !unit vector in direction of earth

            f(1:3) =  y(4:6)
            f(4:6) =  - gravity * mass_sun / r_e**2  * r_e_hat



            !---------------

            k = dt * f

        end function kv

end subroutine rk4step
```

Figure 4: Runge-Kutte ODE Solver

2 Solutions

2.1 Double Pendulum

The main task was to derive the equations of motion for this system. From Figure 1 we see

$$x_1 = l_1 sin(\theta_1)$$

$$y_1 = -l_1 cos(\theta_1)$$

$$x_2 = l_1 sin(\theta_1) + l_2 sin(\theta_2)$$

$$y_2 = -l_1 cos(\theta_1) - l_2 cos(\theta_2)$$

The total energy is $E = T + V$ where T is kinetic energy and V is potential. Taking the time derivative of the Cartesian equations and substituting into the kinetic and potential energy equations

$$T = 1/2 m_1 v_1^2 + 1/2 m_2 v_2^2$$

$$V = m_1 g y_1 + m_2 g y_2$$

We get both total energy E and the langrangian L

$$E = 1/2 m_1 l_1^2 \dot{\theta_1^2} + 1/2 m_2 [l_1^2 \dot{\theta_1^2} + l_2^2 \dot{\theta_2^2} + 2 l_1 l_2 \dot{\theta_1} \dot{\theta_2} cos(\theta_1 - \theta_2)] - [(m_1 + m_2) g l_1 cos(\theta_1) + m_2 g l_2 cos(\theta_2)]$$

$$L = 1/2 m_1 l_1^2 \dot{\theta_1^2} + 1/2 m_2 [l_1^2 \dot{\theta_1^2} + l_2^2 \dot{\theta_2^2} + 2 l_1 l_2 \dot{\theta_1} \dot{\theta_2} cos(\theta_1 - \theta_2)] + [(m_1 + m_2) g l_1 cos(\theta_1) + m_2 g l_2 cos(\theta_2)]$$

From this Lagrange equation we then derived the Euler-Lagrange for each θ which was input into the rk4 routine.
For θ_1

$$(m_1 + m_2) l_1 \dot{\omega}_1 + m_2 l_2 \dot{\omega}_2 cos(\theta_1 - \theta_2) + m_2 l_2 \dot{\theta_2^2} sin(\theta_1 - \theta_2) + g(m_1 + m_2) sin(\theta_1) = 0$$

For θ_2

$$m_2 l_2 \dot{\omega}_2 + m_2 l_1 \dot{\omega}_1 cos(\theta_1 - \theta_2) - m_2 l_1 \dot{\theta_1^2} sin(\theta_1 - \theta_2) + m_2 g sin(\theta 2) = 0$$

```fortran
module setup

    use numtype
    implicit none

    integer, parameter :: n_eq = 4
    real(dp), parameter :: gravity = 9.8_dp, l_1 = 0.5_dp, &
        mass_1 = 3.0_dp, l_2 = 0.4_dp, mass_2 = 2, &
            alpha_1 = pi/4, alpha_2 = pi/4

end module setup


program double_pendulum

    use setup
    implicit none

    real(dp) :: t, dt,  tmax, eps, delt
    real(dp), dimension(n_eq) :: y

    eps = .01
    delt = .01
    t = 0
    dt = .01                                        !time step
    tmax = 200                                      !200 second observation

    y(1:2)   =  (/ alpha_1, 0._dp /)                !initial (theta 1, omega 1)
    y(3:4)   =  (/ alpha_1, 0._dp /)                !initial (theta 2, omega 2))

    do while ( t < tmax )

        write(39,*) l_1 * sin(y(1)), -l_1 * cos(y(1))
        write(40,*) l_1 * sin(y(1)) + l_2 * sin(y(3)), -l_1 * cos(y(1)) - l_2 * cos(y(3))
        write(71,*) t, dt

        if ((y(1) - eps) < 0 .AND. y(2) > 0) then
            write(41,*) y(3), y(4)
        endif

        call rk4step ( t, dt, y )

    end do


end program double_pendulum
```

Figure 5: Code File for Double Pendulum

```fortran
function kv(t, dt, y ) result(k)

    use setup
    implicit none
    real(dp), intent(in) :: t, dt
    real(dp), dimension(n_eq), intent(in) :: y
    real(dp), dimension(n_eq) :: f, k

    !--------------------


    f(1) =  (-gravity*(2*mass_1 + mass_2)*sin(y(2))-mass_2*gravity*sin(y(2)-2*y(4))-2*sin(y(2) &
        -y(4))*mass_2*((y(3))**2*l_2 + (y(1))**2*l_1*cos(y(2) - y(4))))/ &
            (l_1*(2*mass_1 + mass_2 - mass_2*cos(2 * y(2) - 2 * y(4))))

    f(2) = y(1)


    f(3) = (2*sin(y(2)-y(4))*((y(1))**2*l_1*(mass_1+mass_2)*cos(y(2)) + &
        (y(3))**2*l_2*mass_2*cos(y(2)-y(4))))/(l_2*(2*mass_1+mass_2-mass_2*cos(2*y(2)-2*y(4))))

    f(4) = y(3)



    !---------------

    k = dt * f

end function kv
```

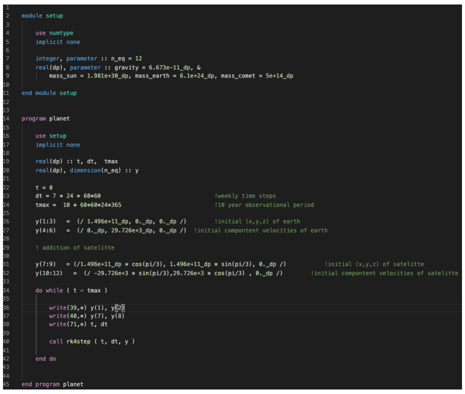
Figure 6: Adjusted section of rk4 for Pendulum

2.2 Sun, Earth, Satellite

A ten year observational period with weekly time steps were used, and the rk4 force vectors were adjusted to incorporate the satellite with the gravitational effects of both the Sun and Earth. The effect of the satellite on both Earth and Sun were neglected as well as the Earth's pull on the Sun. The Lagrange points were calculated and input as initial conditions to be modeled.

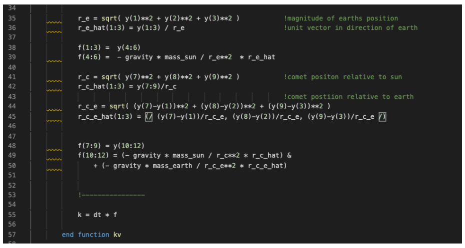
- L1 was calculated using 0.99AU
- L2 was calculated using 1AU + 1.5e9m
- L4 used 1AU forming and equilateral triangle with Sun and Earth

The velocity was then calculated by $v_{earth}/r_{earth} = v_{sat}/r_{sat}$ as a stationary orbit is defined as one at which the body orbits with the same period as Earth. An L5 position exists just opposite L4, lagging behind earth rather than in front. L3 was not calculated.



```fortran
module setup

    use numtype
    implicit none

    integer, parameter :: n_eq = 12
    real(dp), parameter :: gravity = 6.673e-11_dp, &
        mass_sun = 1.981e+30_dp, mass_earth = 6.1e+24_dp, mass_comet = 5e+14_dp

end module setup


program planet

    use setup
    implicit none

    real(dp) :: t, dt,  tmax
    real(dp), dimension(n_eq) :: y

    t = 0
    dt = 7 * 24 * 60*60                                  !weekly time steps
    tmax =  10 * 60*60*24*365                            !10 year observational period

    y(1:3)   =  (/ 1.496e+11_dp, 0._dp, 0._dp /)        !initial (x,y,z) of earth
    y(4:6)   =  (/ 0._dp, 29.726e+3_dp, 0._dp /)  !initial compontent velocities of earth

    ! addition of satelitte

    y(7:9)   = (/1.496e+11_dp * cos(pi/3), 1.496e+11_dp * sin(pi/3), 0._dp /)          !initial (x,y,z) of satelitte
    y(10:12)   =  (/ -29.726e+3 * sin(pi/3),29.726e+3 * cos(pi/3) , 0._dp /)       !initial compontent velocities of satelitte

    do while ( t < tmax )

        write(39,*) y(1), y(2)
        write(40,*) y(7), y(8)
        write(71,*) t, dt

        call rk4step ( t, dt, y )

    end do


end program planet
```

Figure 7: Main Earth, Sun Satellite code file



```fortran
            r_e = sqrt( y(1)**2 + y(2)**2 + y(3)**2 )          !magnitude of earths position
            r_e_hat(1:3) = y(1:3) / r_e                        !unit vector in direction of earth

            f(1:3) =  y(4:6)
            f(4:6) =  - gravity * mass_sun / r_e**2  * r_e_hat

            r_c = sqrt( y(7)**2 + y(8)**2 + y(9)**2 )          !comet positon relative to sun
            r_c_hat(1:3) = y(7:9)/r_c
                                                               !comet postiion relative to earth
            r_c_e = sqrt( (y(7)-y(1))**2 + (y(8)-y(2))**2 + (y(9)-y(3))**2 )
            r_c_e_hat(1:3) = (/ (y(7)-y(1))/r_c_e, (y(8)-y(2))/r_c_e, (y(9)-y(3))/r_c_e /)


            f(7:9) = y(10:12)
            f(10:12) = (- gravity * mass_sun / r_c**2 * r_c_hat) &
                + (- gravity * mass_earth / r_c_e**2 * r_c_e_hat)


            !---------------

            k = dt * f

        end function kv
```

Figure 8: Adjusted section of rk4 file for satelitte

2.3 Driven Mass-Spring

This system was solved numerically using the momentum of mass and the spring force as shown in Figure 5. The motion of the suspension point was parameterized with respect to time. The spring force was then calculated in each time step using the relative position of the mass to the suspension point.

```fortran
module setup

    use numtype
    implicit none

    real(dp), parameter :: mass = 1.0_dp, g = 9.8_dp, &
        spring_k = 10_dp, length_0 = 2.0_dp, tmax = 30_dp, &
        r_sus = 0.5_dp


end module setup


program mass_spring

    use setup
    implicit none

    real(dp) :: t, dt, dx, dy, mag_l, stretch
    real(dp), dimension(3) :: bob_pos, bob_vel, bob_mom, F_spring, F_g, &
        F, delta, l, l_hat

    t = 0
    dt = .01

    bob_pos = (/3.5_dp, 0._dp, 0._dp/)              !initial position of bob
    bob_vel = (/0._dp, 0._dp, 0._dp/)               !initial veloctiy of bob
    bob_mom = mass * bob_vel

    !F_g = (/0._dp, -g*mass, 0._dp/)



    do while (t < tmax)

        dx = r_sus * cos(7*t)                       !time parameterization
        dy = r_sus * sin(7*t)                       !of suspension point

        delta = (/dx, dy, 0._dp/)                   !position of sus point at time t

        l = bob_pos - delta
        mag_l = sqrt(l(1)**2 + l(2)**2 + l(3)**2)
        l_hat = l/mag_l
        stretch = mag_l - length_0

        F_spring = -spring_k * stretch * l_hat
        F = F_spring! + F_g

        bob_mom = bob_mom + F * dt
        bob_vel = bob_mom/mass
        bob_pos = bob_pos + bob_vel * dt

        write(13,*) bob_pos(1), bob_pos(2)
        write(14,*) t

        t = t + dt
    end do
end program mass_spring
```

Figure 9: Numerical Solution to Driven Spring System

3 Results

3.1 Double Pendulum

At small angles the behavior was simple. The chaotic state increased with increased *theta*. An IF statement was used to plot ω_2 as a function of θ_2 when $\theta_1 = 0$ and $\omega_1 > 0$.

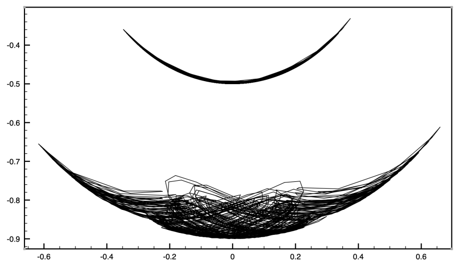


Figure 10: Position of both masses Pi/4 I.C.

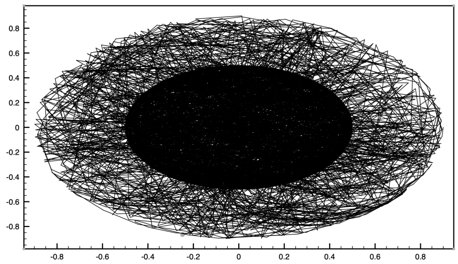


Figure 11: Position of both masses Pi/2 I.C.

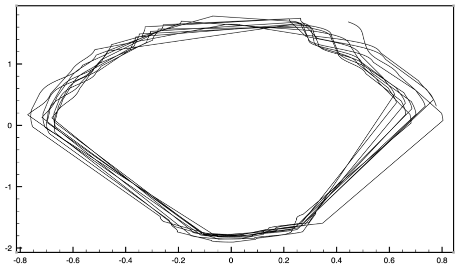


Figure 12: θ_2 vs. ω_2

3.2 Sun, Earth, Satellite

The first Satellite orbit generated was heavily elliptical.

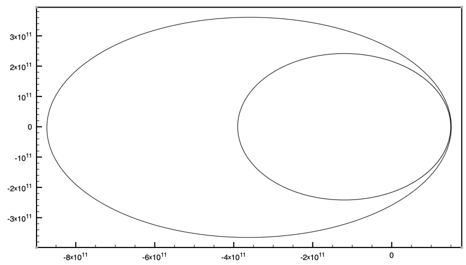


Figure 13: Satellite at arbitrary distance and speed

L1 and L2 had a narrow parameter for which they behaved as a stationary orbit.

The L4 and L5 positions matched Earth's orbit.

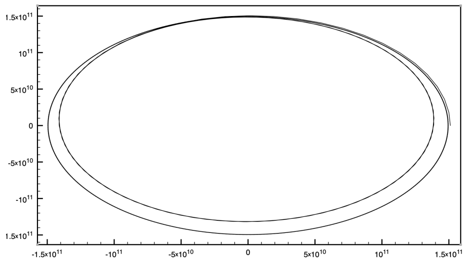


Figure 14: L2 Lagrange point orbit

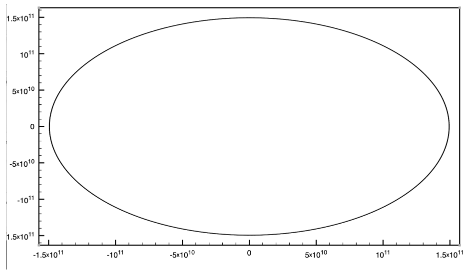


Figure 15: L4 orbit overlapping Earth's

3.3 Driven Mass-Spring

The resulting position of the mass was plotted in the x-y plane for a total of 30 seconds.

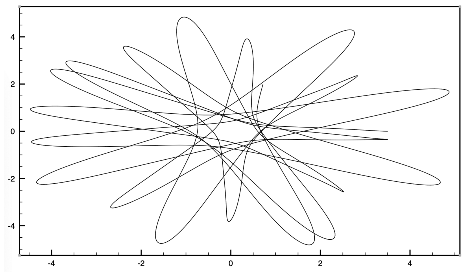


Figure 16: Position of Mass in x-y plane