HW5 Simple Dynamical Systems

Fei Ding

November 16, 2009

Problem1

For one dimensional simple harmonic oscillator, the equation of motion is

$$m\frac{dv}{dt} = \frac{F}{m} = \frac{-kx}{m}$$

$$\frac{dx}{dt} = v$$

According to Euler Method,

$$\begin{aligned} v_{i+1} &= v_i - \frac{kx}{m}\,\Delta t \\ x_{i+1} &= x_i + v_i\,\Delta t \end{aligned} \tag{1}$$

And the expression for energy,

$$\begin{aligned} E_i &= \frac{1}{2}m{v_i}^2 + \frac{1}{2}kx_i^2 \\ E_{i+1} &= \frac{1}{2}m{v_{i+1}}^2 + \frac{1}{2}kx_{i+1}^2, \end{aligned} \tag{2}$$

Plug eq(1) into eq(2), get

$$\begin{aligned} E_{i+1} &= \frac{1}{2}m(v_i - \frac{kx_i}{m}\,\Delta t)^2 + \frac{1}{2}k(x_i + v_i\,\Delta t)^2 \\ &= \frac{1}{2}m({v_i}^2 + \frac{k^2{x_i}^2}{m}\Delta t^2 - 2\frac{kx_i}{m}v_i\Delta t) + \frac{1}{2}k({x_i}^2 + {v_i}^2\Delta t^2 + 2x_iv_i\Delta t) \end{aligned}$$

So

$$E_{i+1} - E_i = (\frac{1}{2}m{v_i}^2 + \frac{1}{2}kx_i^2)\frac{k}{m}\Delta t^2$$

and

$$\delta E = \frac{E_{i+1} - E_i}{E_i} = \frac{k}{m}\Delta t^2 > 0$$

$\delta E > 0$ is always true. So the Euler method does not conserve energy for simple harmonic oscilallator.

Problem2

The equations of motion for a golf ball are

$$m\frac{d\vec{v}}{dt} = \vec{F}_{mag} + \vec{F}_{grav} + \vec{F}_{drag}$$

$$\frac{d\vec{x}}{dt} = \vec{v}$$

Apply Euler method to solve the problem in x and y direction.

Considering all the three forces, I got the ranges and total time for different initial angles as following,

θ_0=7° R=216.51m t=7.942s

θ_0=9° R=215.48m t=8.260s

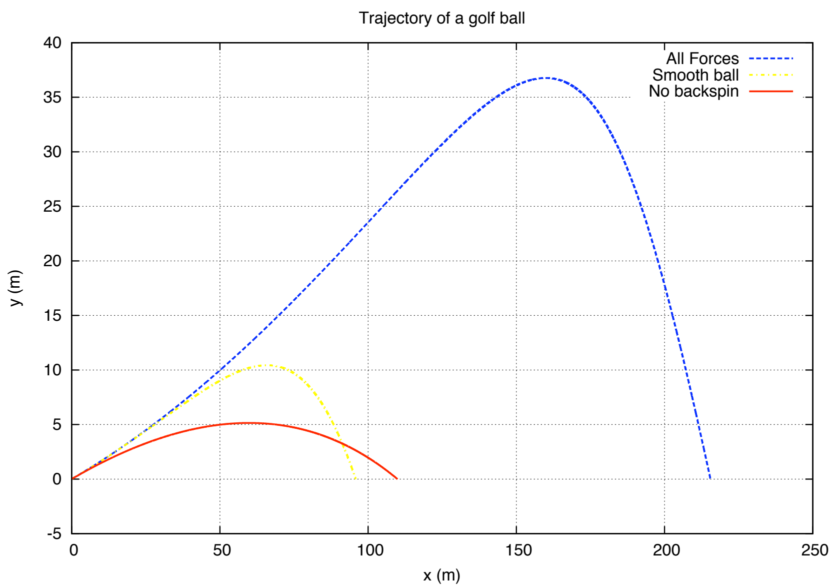
θ_0=11° R=213.80m t=8.547s

The trajectories for a golf ball with initial angle $\theta_0 = 9°$ while considering it as a normal ball with all the three forces, a smooth ball with C=1/2 and a ball without backspin are shown below.

The normal ball can reach the highest height and longest range. and somehow shows a curve against the gravitational force for the rising part.

For the smooth ball, the dragging constant becomes a constant 1/2 rather than dependent on the speed of the ball. From the expression of dragging constant it can be seen that while $v > v_c$ the constant would be smaller than 1/2, so the dragging force would be smaller. That's why the smooth ball cannot fly as high as the normal ball (Here, v_c is set to be 14 m/s, which is low enough to be reached).

The ball without backspin shown a trajectory more or less like a ball only affected by gravitational force, however, the rising and falling part are not symmetry.



Problem3

The equations of motion for a driven, damped oscillator are(with the parameters provided in this problem)

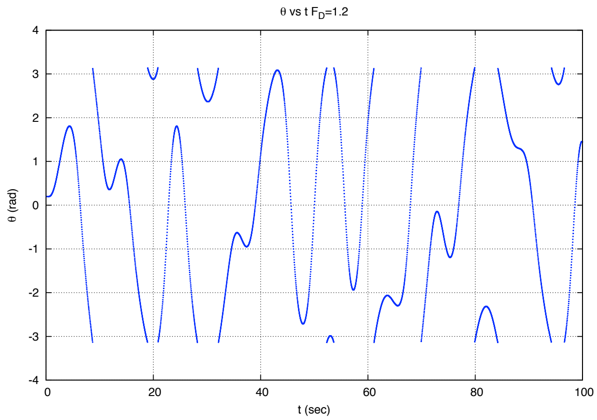
$$\frac{d\omega}{dt} = -sin\theta - \frac{1}{2}\omega + F_D sin\frac{2}{3}t$$

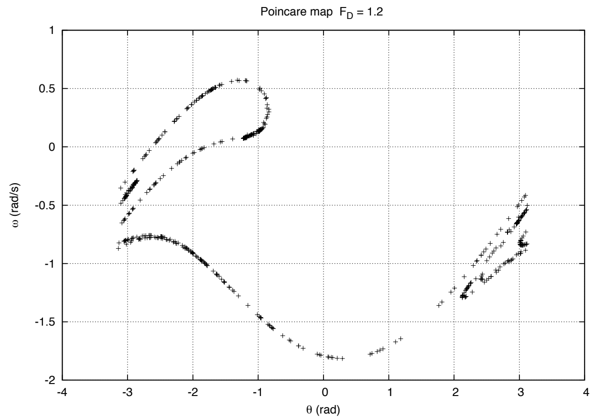
$$\frac{d\theta}{dt} = \omega$$

Apply RK4 method to solve the problem.

1)For $F_D = 1.2$, the chaos can be seen clearly from the θ vs. t plot.

At time $t = nT$, where $T = 2\pi/\Omega_D$ is the period of the driving force, plot ω vs. θ, the plot agrees with what is given in the lecture. It's a plot with complex structure rather than just a single point.



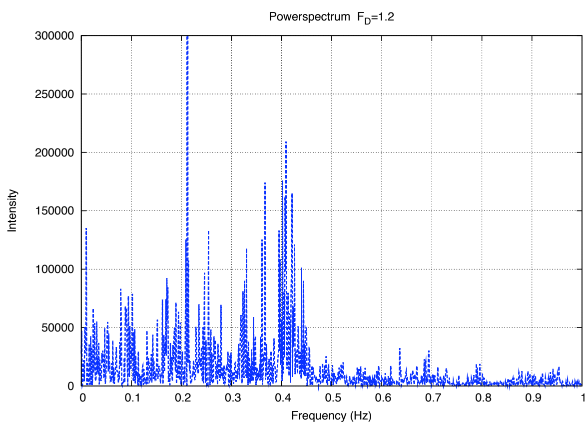


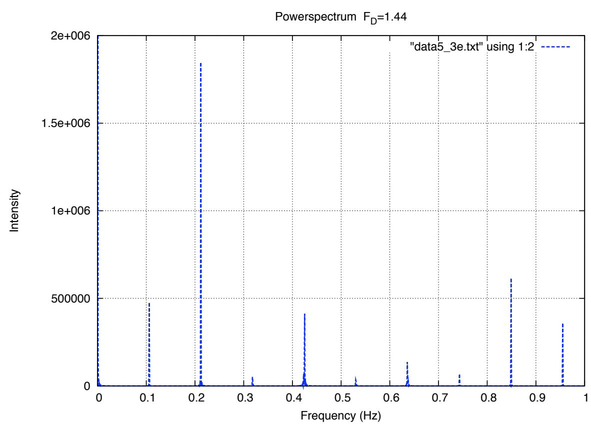
Then I did Fourier transformation to find the power spectrum of frequency.

$$\tilde{\theta}(f) = \int_{-\infty}^{+\infty} \theta(t) e^{-i\pi f t}\, dt = \int_{-\infty}^{+\infty} \theta(t) \cos \pi f t\, dt - i \int_{-\infty}^{+\infty} \theta(t) \sin \pi f t\, dt$$

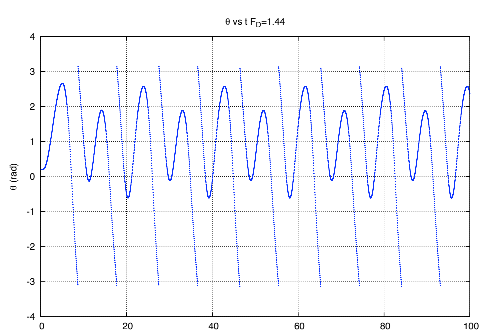
$$PS(f) = |\tilde{\theta}(f)|^2 = \left(\int_{-\infty}^{+\infty} \theta(t) \cos \pi f t\, dt \right)^2 + \left(\int_{-\infty}^{+\infty} \theta(t) \sin \pi f t\, dt \right)^2$$

The power spectrum of $F_D = 1.2$ has a lot of gigs because basically this case is chaotic case.





2)For $F_D = 1.44$, the θ vs. t plot shows a "two periods" patten.



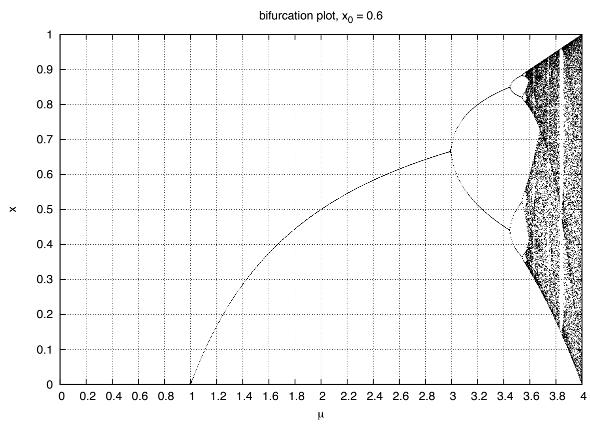
The power spectrum of $F_D = 1.44$ is more simple with several peaks, no gigs.

Problem4

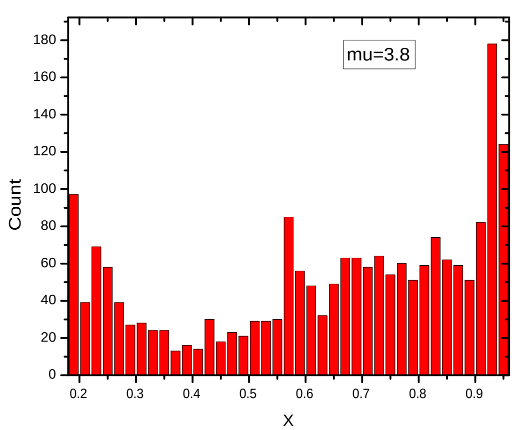
The logistics map, $x_{n+1} = f(x_n)$, $f(x) = \mu x(1-x)$.

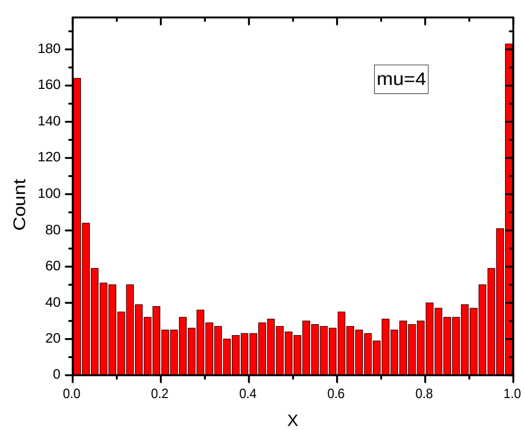
I set the seed $x_0 = 0.6$, got a data file for $\mu, x, f(x)$, then used Excel to delete those repeated ones and got a new data file then plot x vs. μ, I got the bifurcation plot.

The bifurcation plot shows that while μ is less than 1 the x is all around zero, that's because μ, x and $(1-x)$ are all smaller than 1 so their product $f(x)$ can only be smaller than 1 and got smaller and smaller until finally fell to zero. From $\mu = 1$ to $\mu = 3$ there is only one x, so it is stable. The first bifurcation shown around $\mu = 3$ and more and more bifurcation for larger μ, which corresponds to unstable part.



The invariant density for $\mu = 3.8$ and $\mu = 4$ are shown below. The histogram of $\mu = 4$ corresponds to the analytical result $\rho(x) = \frac{1}{\pi\sqrt{x(1-x)}}$ well, while the histogram of $\mu = 3.8$ is not so smooth in the middle.





Codes

Problem2

```
MODULE parameters
  IMPLICIT NONE
  SAVE
  DOUBLEPRECISION, PARAMETER :: pi = 3.1415926535897932384626433832795028841971693993751
  DOUBLEPRECISION, PARAMETER :: dt = 1D-3
  DOUBLEPRECISION, PARAMETER :: mass = 0.046D0
  DOUBLEPRECISION, PARAMETER :: rho = 1.2D0, A = 0.00143D0 !air density and cross section

END MODULE parameters

PROGRAM GOLF
 USE parameters
 IMPLICIT NONE

 INTEGER :: n
 DOUBLEPRECISION :: t !the time of now
 DOUBLEPRECISION, PARAMETER :: tmax = 10D0 ! in sec
 DOUBLEPRECISION :: x, xp, vx, vxp, y, yp, vy, vyp !x for current x, xp for previous x, same for vx, y a
 DOUBLEPRECISION :: range
 DOUBLEPRECISION :: theta0, v0 !initial angel, initial velocity

 OPEN(unit=14, file='5_2c.txt')

 PRINT*, "Input launching angle:"
 READ*, theta0

 x = 0D0
 y = 0D0
 v0=70d0

 vx = v0*DCOS(theta0*pi/180D0)
 vy = v0*DSIN(theta0*pi/180D0)

 DO n = 1, NINT(tmax/dt)
   t = n*dt

   !x-algorithm
   xp = x
   vxp = vx
   yp = y
   vyp = vy
   x = xp + vxp * dt
   vx = vxp + (fgravx(xp, vxp) + fDragx(vxp, vyp) + fMagx(vxp, vyp) )/mass * dt
   !for ball without backspin
   !vx=vxp + (fgravx(xp, vxp) + fDragx(vxp, vyp) )/mass * dt

   !y-algorithm
   y = yp + vyp * dt
   vy = vyp + (fgravy(yp, vyp) + fDragy(vxp, vyp) + fMagy(vxp, vyp) )/mass * dt
```

```
   ! for ball without backspin
   !vy = vyp + (fgravy(yp, vyp) + fDragy(vxp, vyp) )/mass * dt
   WRITE(14,'(5F15.9)') t, x, y, vx, vy
  IF (y < 0D0) THEN
    PRINT*, "The golf ball hits the ground at t=", t , " sec,"
PRINT*, "Range = ", x
    EXIT
   ENDIF
  ENDDO

 CONTAINS

!External forces functions
 DOUBLEPRECISION FUNCTION fgravx(x,vx)
  IMPLICIT NONE
  DOUBLEPRECISION :: x, vx
  fgravx = 0D0
  RETURN
 END FUNCTION

 DOUBLEPRECISION FUNCTION fgravy(y,vy)
  USE parameters
  IMPLICIT NONE
  DOUBLEPRECISION :: y, vy
  DOUBLEPRECISION, PARAMETER :: g = 9.8D0
  fgravy = mass*(-g)
  RETURN
 END FUNCTION

 DOUBLEPRECISION FUNCTION fdragx(vx,vy)
  USE parameters
  IMPLICIT NONE
  DOUBLEPRECISION :: vx, vy, v !x-component, y-component and magnitude of v
  DOUBLEPRECISION :: theta !direction of motion
  theta = DATAN(vy/vx)
  v = DSQRT(vx**2+vy**2)
  fDragx = -dragconst(v) * rho * A * (v**2) * DCOS(theta)
  RETURN
 END FUNCTION

 DOUBLEPRECISION FUNCTION fdragy(vx,vy)
  USE parameters
  IMPLICIT NONE
  DOUBLEPRECISION :: vx, vy, v
  DOUBLEPRECISION :: theta
  theta = DATAN(vy/vx)
  v = DSQRT(vx**2+vy**2)
  fDragy = -dragconst(v) * rho * A * (v**2) * DSIN(theta)
  RETURN
 END FUNCTION

 DOUBLEPRECISION FUNCTION fMagx(vx,vy)
   USE parameters
```

```
   IMPLICIT NONE
   DOUBLEPRECISION :: vx, vy
   DOUBLEPRECISION, PARAMETER :: beta = 0.25D0 !beta = (S0 * omega/mass)
   fMagx = - beta * mass * vy
   RETURN
 END FUNCTION

 DOUBLEPRECISION FUNCTION fMagy(vx,vy)
   USE parameters
   IMPLICIT NONE
   DOUBLEPRECISION :: vx, vy
   DOUBLEPRECISION, PARAMETER :: beta = 0.25D0 !beta = (S0 * omega/mass)
   fMagy =  beta * mass * vx
   RETURN
 END FUNCTION

 DOUBLEPRECISION FUNCTION dragconst(v)
   IMPLICIT NONE
   DOUBLEPRECISION, PARAMETER :: vc = 14D0 !critical transient speed
   DOUBLEPRECISION :: v
   !for smooth ball
   !dragconst=0.5d0
   IF (v < vc) THEN
     dragconst = 0.5D0
   ELSE
     dragconst = 0.5D0*vc/v
   ENDIF
   RETURN
 END FUNCTION
END PROGRAM
```

Problem3

```
MODULE parameters
  IMPLICIT NONE
  SAVE
  DOUBLEPRECISION, PARAMETER :: pi = 3.14159265358979323846264338328
  DOUBLEPRECISION, PARAMETER :: fd = 1.2D0 ! co-eff of driving force term
  DOUBLEPRECISION, PARAMETER :: omegaD =2d0/3d0    !0.66666666666666666666D0
  DOUBLEPRECISION, PARAMETER :: tp = 3d0*pi !period=(2 Pi/omegaD)
  DOUBLEPRECISION, PARAMETER :: dt = tp/300d0 !so we can keep track at time integer multiple of tp
 END MODULE parameters

PROGRAM pendulum
 USE parameters
 IMPLICIT NONE
 !INTEGER :: ierr
 INTEGER :: i
 DOUBLEPRECISION :: t
 DOUBLEPRECISION, PARAMETER :: tmax = 5000D0 ! sec
 DOUBLEPRECISION :: theta, omega
 DOUBLEPRECISION, DIMENSION(INT(tmax/dt)) :: thetaarray
```

```
 DOUBLEPRECISION :: frequency

 OPEN(unit=14, file='data5_3a.txt')
 OPEN(unit=21, file='data5_3b.txt')
 OPEN(unit=15, file='data5_3c.txt')
 theta = 0.2D0
 omega = 0D0

 DO i = 1, INT(tmax/dt)
   t = i*dt
   CALL rk4(t, theta, omega, fx, fvx)

   !make theta periodic
   IF (theta > pi) THEN
     theta = theta - 2D0*pi
   ELSEIF (theta < -pi) THEN
     theta = theta + 2D0*pi
   ELSE
     theta = theta
   ENDIF
   thetaarray(i) = theta
   WRITE(14,'(5F15.9)') t, theta, omega

!data for Poincare map
   IF ( MOD(i, 300) == 0) THEN  !record the point at t=n*period
    WRITE(21, *) t, theta, omega
   ENDIF
 ENDDO

!print data for power spectrum
 DO frequency = 0D0, 1D0, 0.001D0
   WRITE(15, *) frequency, powerSpect(thetaarray, frequency)
 ENDDO
 CONTAINS

!the differentiation equation for theta d theta/dt = omega
 DOUBLEPRECISION FUNCTION fx(t, theta, omega)
   USE parameters
   IMPLICIT NONE
   DOUBLEPRECISION :: t, theta, omega
   fx = omega
   RETURN
 END FUNCTION

!the differentiation equation for omega d omega/dt = fvx
 DOUBLEPRECISION FUNCTION fvx(t, theta, omega)
   USE parameters
   IMPLICIT NONE
   DOUBLEPRECISION :: t, theta, omega
   DOUBLEPRECISION :: fHook, fdrag, fdrive
   fvx = -DSIN(theta)-0.5d0*omega+fd* DSIN(omegaD * t)
   RETURN
 END FUNCTION
```

```
 SUBROUTINE rk4(t, x, vx, fx, fvx)
   USE parameters
   EXTERNAL fx, fvx
   DOUBLEPRECISION :: fx, fvx
   DOUBLEPRECISION :: t, x, vx  ! x and vx are both inputs and outputs. the values will be updated
   DOUBLEPRECISION :: k1x, k2x, k3x, k4x
   DOUBLEPRECISION :: k1vx, k2vx, k3vx, k4vx
   k1x  = fx(t, x, vx) * dt
   k1vx = fvx(t, x, vx)* dt
   k2x  = fx(t+0.5D0*dt, x+0.5D0*k1x, vx+0.5D0*k1vx) * dt
   k2vx = fvx(t+0.5D0*dt, x+0.5D0*k1x, vx+0.5D0*k1vx)* dt
   k3x  = fx(t+0.5D0*dt, x+0.5D0*k2x, vx+0.5D0*k2vx) * dt
   k3vx = fvx(t+0.5D0*dt, x+0.5D0*k2x, vx+0.5D0*k2vx)* dt
   k4x  = fx(t+dt, x+ k3x, vx+k3vx) * dt
   k4vx = fvx(t+dt, x+ k3x, vx+k3vx) * dt
   x = x + (k1x + 2D0*k2x  + 2D0*k3x  + k4x)/6D0
   vx=vx + (k1vx+ 2D0*k2vx + 2D0*k3vx + k4vx)/6D0
END SUBROUTINE

 DOUBLEPRECISION FUNCTION powerSpect(array,f)
   USE parameters
   IMPLICIT NONE
   INTEGER :: i
   DOUBLEPRECISION :: f
   DOUBLEPRECISION :: FReal, FIm
   DOUBLEPRECISION, DIMENSION(INT(tmax/dt)) :: weight
   DOUBLEPRECISION, DIMENSION(INT(tmax/dt)) :: array

!Integration by trapzoid method
  !generating the weights
  DO i = 1, INT(tmax/dt) !dtermining tpart and weight
     IF (i ==1 .OR. i == INT(tmax/dt)) THEN
       weight(i) = 0.5D0
     ELSE
       weight(i) = 1D0
     ENDIF
  ENDDO

 !Real part integral
  FReal = 0D0
  DO i = 1, INT(tmax/dt)
   FReal = FReal + array(i)*DCOS(pi*f*(i*dt))*weight(i)
  ENDDO
   FReal = FReal*dt

  !Imaginary part integral
  FIm = 0D0
  DO i = 1, INT(tmax/dt)
   FIm = FIm + array(i)*DSIN(pi*f*(i*dt))*weight(i)
  ENDDO !tindex
   FIm = FIm*dt
   powerSpect = FReal**2 + FIm**2
```

```
  RETURN
END FUNCTION
END PROGRAM
```

Problem4

```
 PROGRAM logisticsmap
 IMPLICIT NONE
 INTEGER :: i
 DOUBLEPRECISION :: x
 DOUBLEPRECISION :: mu !growth parameter
 OPEN(unit=14, file='data5_4.txt')
 DO mu = 0D0, 4D0, 0.005D0
! for mu=3.8 and 4.0 case
!   print *, "Input mu:"
!   read(*,*) mu
   x = 0.6D0
   DO i = 1, 1000  !Wait until the dust settles
     x = func(mu, x)
   ENDDO
   DO i = 1, 2000 !start to print the value of mu and x
     x = func(mu, x)
 WRITE(14, '(F8.5)') x
   ENDDO
   print *, "mu=", mu
 ENDDO
 CONTAINS

 DOUBLEPRECISION FUNCTION func(mu, x)
  IMPLICIT NONE
  DOUBLEPRECISION :: x, mu
  func = mu * x * (1D0 - x)
  RETURN
 END FUNCTION
END PROGRAM
```