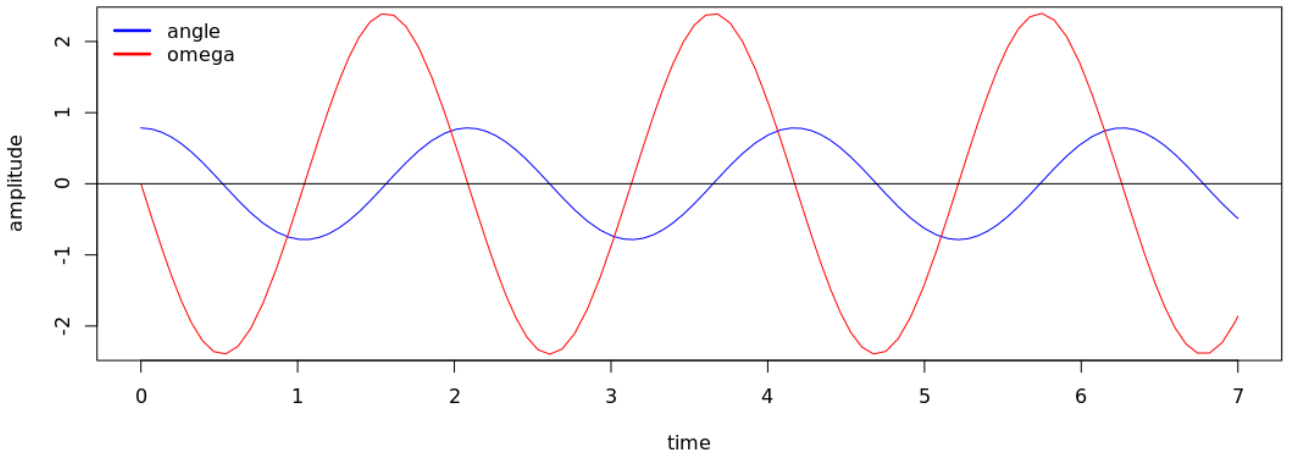
CS-7863: Scientific and Statistical Computing
Homework #4
Noah L. Schrick – 1492657

1.)

```
## 1. Transform the second order ODE into a system of two first order ODEs
# use ode45 or rk4sys to solve for the motion of the plane pendulum
if (!require("pracma")) install.packages("pracma")
library(pracma)

tmin <- 0
tmax <- 7
theta0 <- pi/4
omega0 <- 0
y.init <- c(theta0, omega0)
g <- 9.81
L <- 1.0

## Substitution Tricks
# theta_1 = theta_prime
# theta_2 = theta
# theta_1_prime = theta_pp
# theta_2_prime = theta_prime = theta_1

## Equation Replacements
ode_1.f <- function(t, y, k){
  theta <- y[1]
  omega <- y[2]
  g <- k[1]
  L <- k[2]
  omega_p <- -k[1]/k[2] * sin(theta)
  theta_p <- omega
  as.matrix(c(theta_p, omega_p))
}

p.params <- c(g, L)

pend.sol <- ode45(f=function(t,y){ode_1.f(t,y,k=p.params)},
                          y=y.init, t0=tmin, tfinal=tmax)
```

a.)

```
plot(pend.sol$t,pend.sol$y[,1],type="l",col="blue",
     ylim=c(-2.3,2.3), xlab="time",ylab="amplitude")
par(new=T)
lines(pend.sol$t,pend.sol$y[,2],type="l",col="red")
abline(h=0)
legend("topleft", # coordinates, "topleft" etc
       c("angle","omega"), # label
       lty=c(1,1), # line
       lwd=c(2.5,2.5), # weight
       #cex=.8,
       bty="n", # no box
       col=c("blue","red") # color
)
```



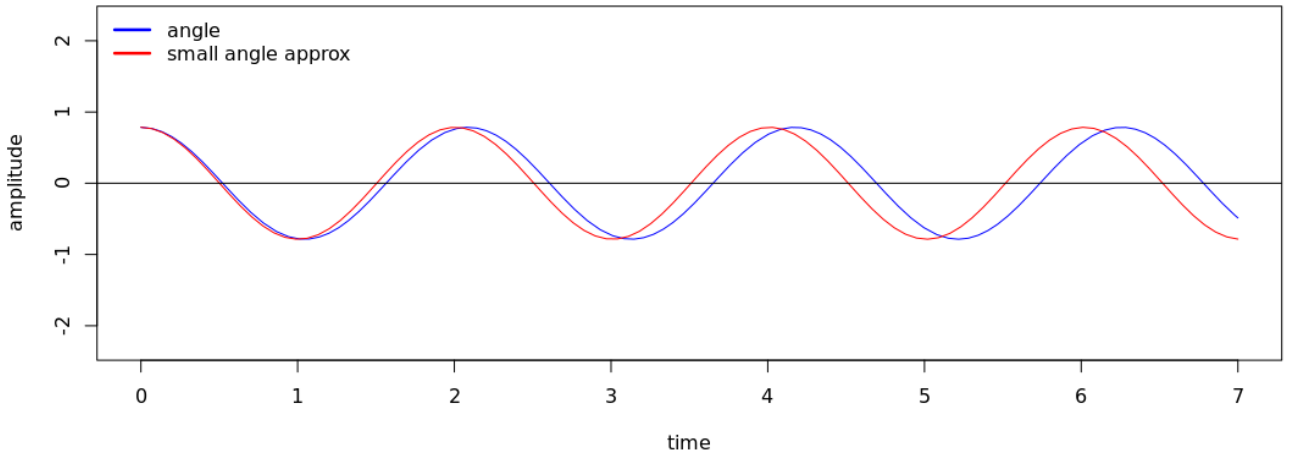
When the plot for the angle crosses the x-axis, what feature occurs in the plot for angular speed?
As the plot for the angle crosses the x-axis, angular speed reaches its minimum or maximum. As the angle goes from positive to negative, angular speed reaches its minimum and begins to increase. As the angle goes from negative to positive, angular speed reaches its maximum and begins to decrease.

When the plot for angular speed crosses the x-axis, what is happening in the plot for the angle?
As the plot for angular speed crosses the x-axis, the angle reaches its minimum or maximum. As angular speed goes from negative to positive, the angle reaches its minimum and begins to increase. As angular speed goes from positive to negative, the angle reaches its maximum and begins to decrease.

What is happening to the motion of the pendulum at these two points and what does it mean in terms of conservation of energy (remember the angular speed is related to kinetic energy)?
When angular speed is at its maximum or minimum the pendulum is at its greatest x value with high kinetic energy and low potential energy. The pendulum immediately after this point starts to swing in the opposite direction, and this is reflected with the change in angle.

b.)

```
# b. Overlay a plot of the small-angle approx
plot(pend.sol$t,pend.sol$y[,1],type="l",col="blue",
     ylim=c(-2.3,2.3), xlab="time",ylab="amplitude")
par(new=T)
lines(seq(tmin,tmax,len=length(pend.sol$t)),

theta0*cos(sqrt(g/L)*seq(tmin,tmax,len=length(pend.sol$t))),type="l",col="red")
abline(h=0)
legend("topleft", # coordinates, "topleft" etc
       c("angle","small angle approx"), # label
       lty=c(1,1), # line
       lwd=c(2.5,2.5), # weight
       #cex=.8,
       bty="n", # no box
       col=c("blue","red") # color
)
```



How does the amplitude of the two solutions differ as time increases?

The amplitude of the two solutions remains the same, but there is a phase shift. As time increases, there starts to be issues where the small-angle approximation strays further from the actual solution due to this phase shift, as described in the following question.

How does the period of the nonlinear solution change in comparison to the small-angle solution as time increases?

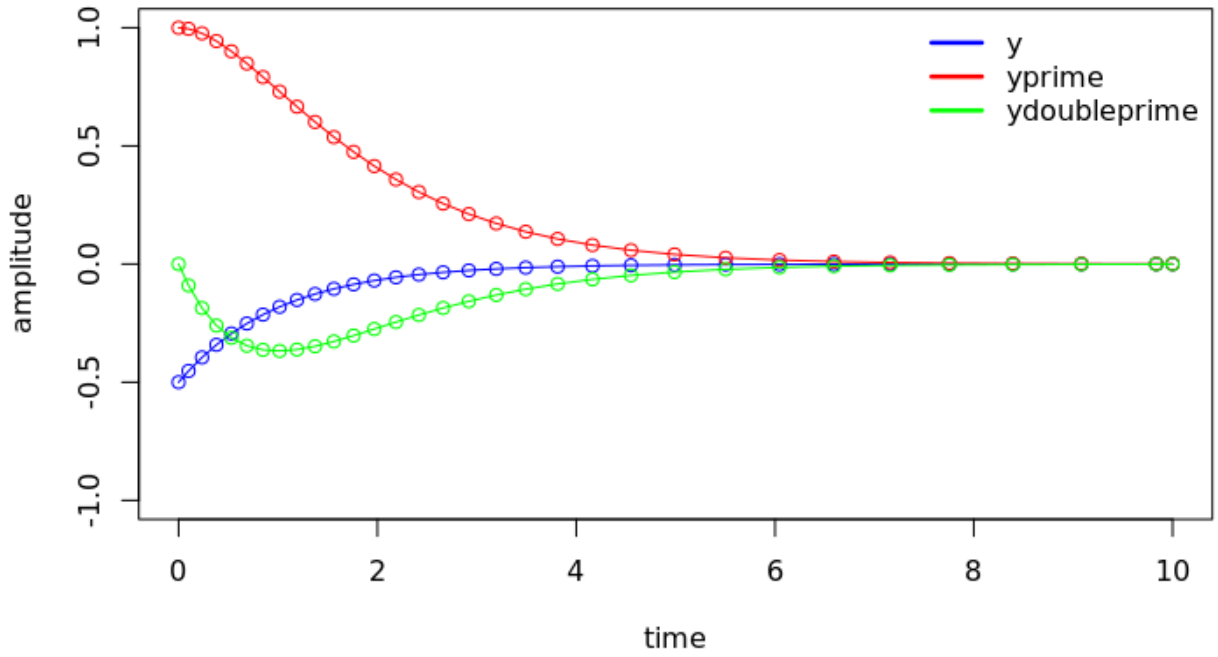
The small-angle solution starts with a period of 2.0, which remains consistent as time increases. The nonlinear solution starts with a period of 2.1, which then increases as time increases. This difference in period is demonstrated through the phase shift.

2.)

```
## 2. Use ode45 to solve the damped harmonic oscillator
tmin <- 0
tmax <- 10
yo <- 1
ybaro <- 0

## Substitution Tricks
# y1 = y_prime
# y_2 = y
# y_1_prime = y_pp
# y_2_prime = y_prime = y_1

# Eqs:
# m*y_1_prime + c*y1 + k*y_2 = 0
# y_1_prime = (-c*y_1 - k*y_2)/m
# y_2_prime = y_1
```

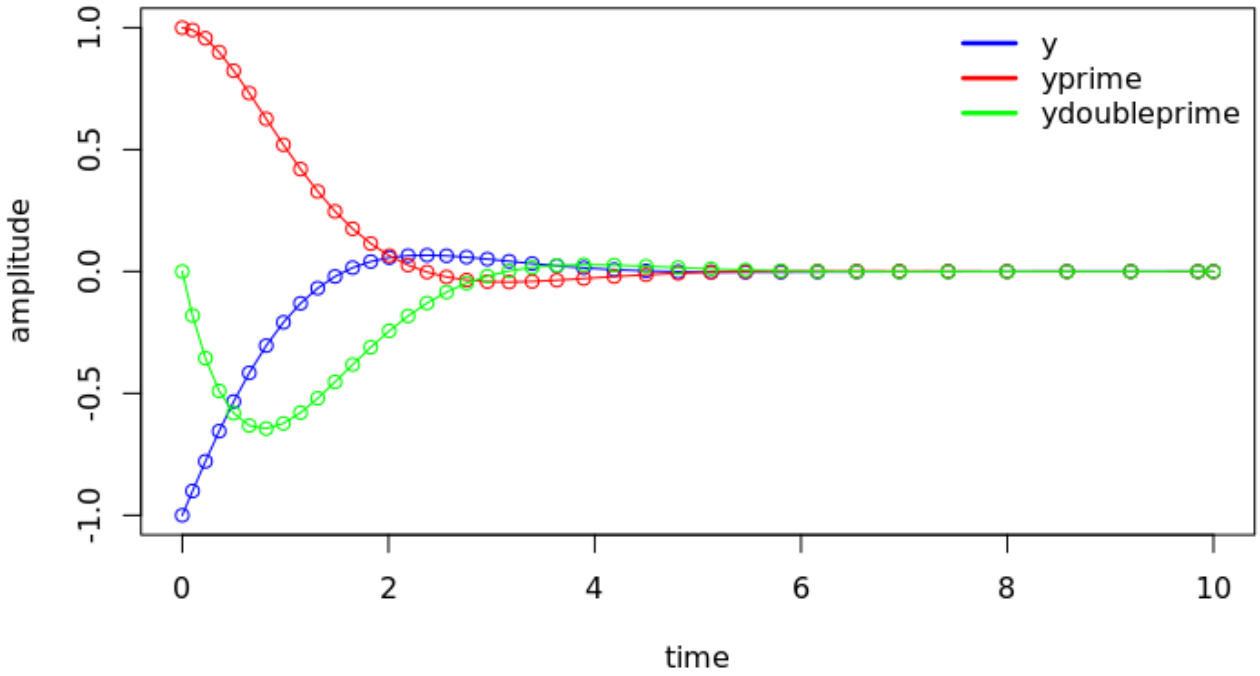
a.)

```
m <- 1
c <- 1
kvar <- 2

dho.f <- function(t, y, kpass){
  yn <- y[1]
  ydot <- y[2]
  #         c               k             m
  ypp <- (-kpass[2]*yn - kpass[3]*ydot)/kpass[1]
  as.matrix(c(ydot,ypp))
}

y.init <- c(yo, ybaro)
dho.params <- c(m,c,kvar)
dho.sol <- ode45(f=function(t,y){dho.f(t,y,kpass=dho.params)},
                 y=y.init, t0=tmin, tfinal=tmax)

# plot pos vs time
pos <- (-m*dho.sol$y[,2] - c*dho.sol$y[,1])/kvar
plot(dho.sol$t, pos ,type="o",col="blue",
     ylim=c(-1.0,1.0), xlab="time",ylab="amplitude")
par(new=T)
lines(dho.sol$t,dho.sol$y[,1],type="o",col="red")
lines(dho.sol$t,dho.sol$y[,2],type="o",col="green")
legend("topright", # coordinates, "topleft" etc
       c("y","yprime", "ydoubleprime"), # label
       lty=c(1,1), # line
       lwd=c(2.5,2.5), # weight
       #cex=.8,
       bty="n", # no box
       col=c("blue","red", "green") # color
)
```



b.)

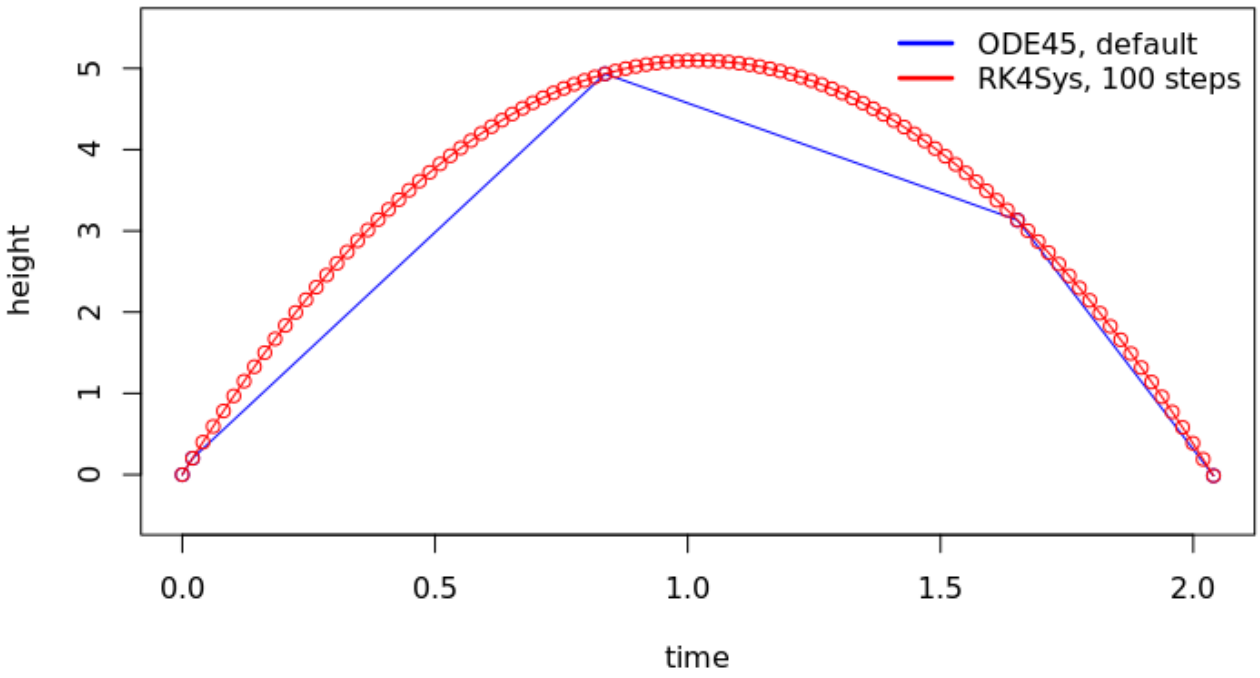
```
m <- 1
c <- 2
k <- 1
dho.params <- c(m,c,k)

y.init <- c(yo, ybaro)
dho.params <- c(m,c,kvar)
dho.sol <- ode45(f=function(t,y){dho.f(t,y,kpass=dho.params)},
                 y=y.init, t0=tmin, tfinal=tmax)

# plot pos vs time
pos <- (-m*dho.sol$y[,2] - c*dho.sol$y[,1])/kvar
plot(dho.sol$t, pos ,type="o",col="blue",
     ylim=c(-1.0,1.0), xlab="time",ylab="amplitude")
par(new=T)
lines(dho.sol$t,dho.sol$y[,1],type="o",col="red")
lines(dho.sol$t,dho.sol$y[,2],type="o",col="green")
legend("topright", # coordinates, "topleft" etc
       c("y","yprime", "ydoubleprime"), # label
       lty=c(1,1), # line
       lwd=c(2.5,2.5), # weight
       #cex=.8,
       bty="n", # no box
       col=c("blue","red", "green") # color
)
```


y
yprime
ydoubleprime
amplitude
time
1.0
0.5
0.0
-0.5
-1.0
0
2
4
6
8
10

3.)

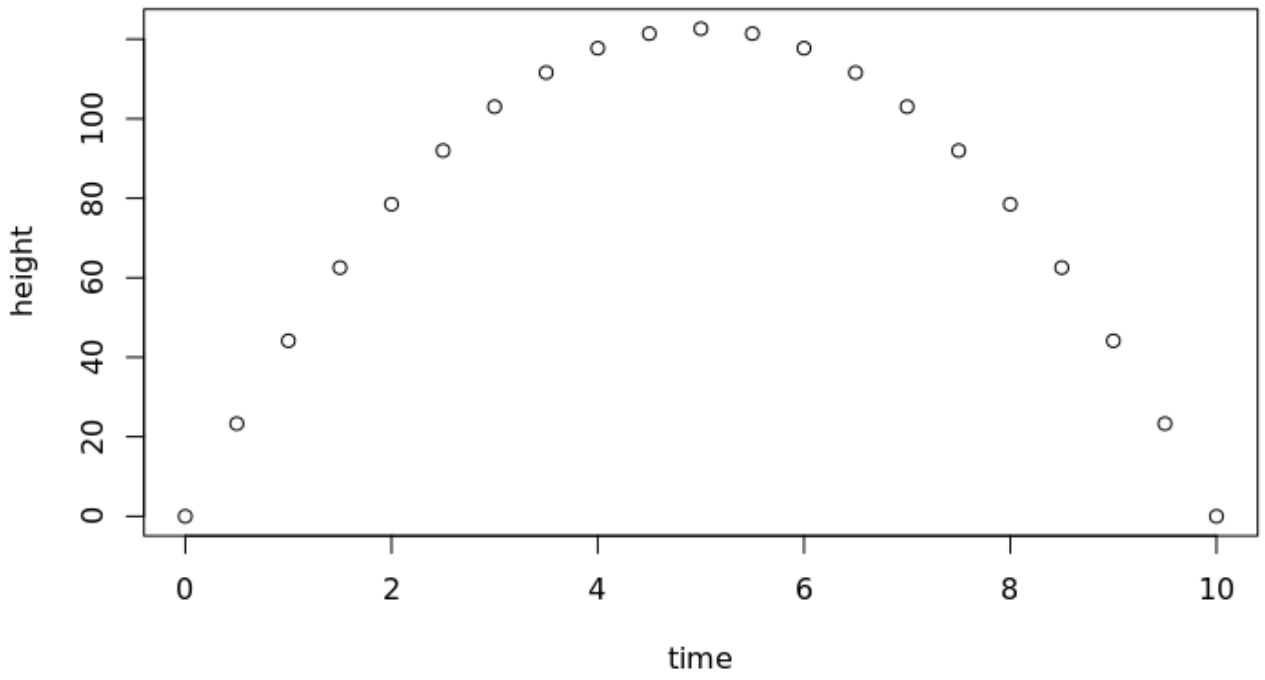
a.)

```
tmin <- 0
tmax <- 2.04
xo <- 0
vo <- 10
projectile.f <- function(t,y){
  g <- 9.81
  v <- y[2]  # y1dot
  a <- -g  # could be any f, y'' = f(t,y)
  matrix(c(v,a))
}

y.init <- c(xo, vo)
ivp.sol <- ode45(f=function(t,y){projectile.f(t,y)},
                 y=y.init, t0=tmin, tfinal=tmax)
ivp.sol.rk4 <- rk4sys(projectile.f, 0, 2.04, c(0, 10), 100)

plot(ivp.sol$t, ivp.sol$y[,1] ,type="o",col="blue",
     ylim = c(-0.5, 5.5), xlab="time",ylab="height")
par(new=T)
lines(ivp.sol.rk4$x, ivp.sol.rk4$y[,1] ,type="o",col="red",
     xlab="time",ylab="height")
legend("topright",
       c("ODE45, default","RK4Sys, 100 steps"), # label
       lty=c(1,1), # line
       lwd=c(2.5,2.5), # weight
       #cex=.8,
       bty="n", # no box
       col=c("blue","red", "green") # color
)
```



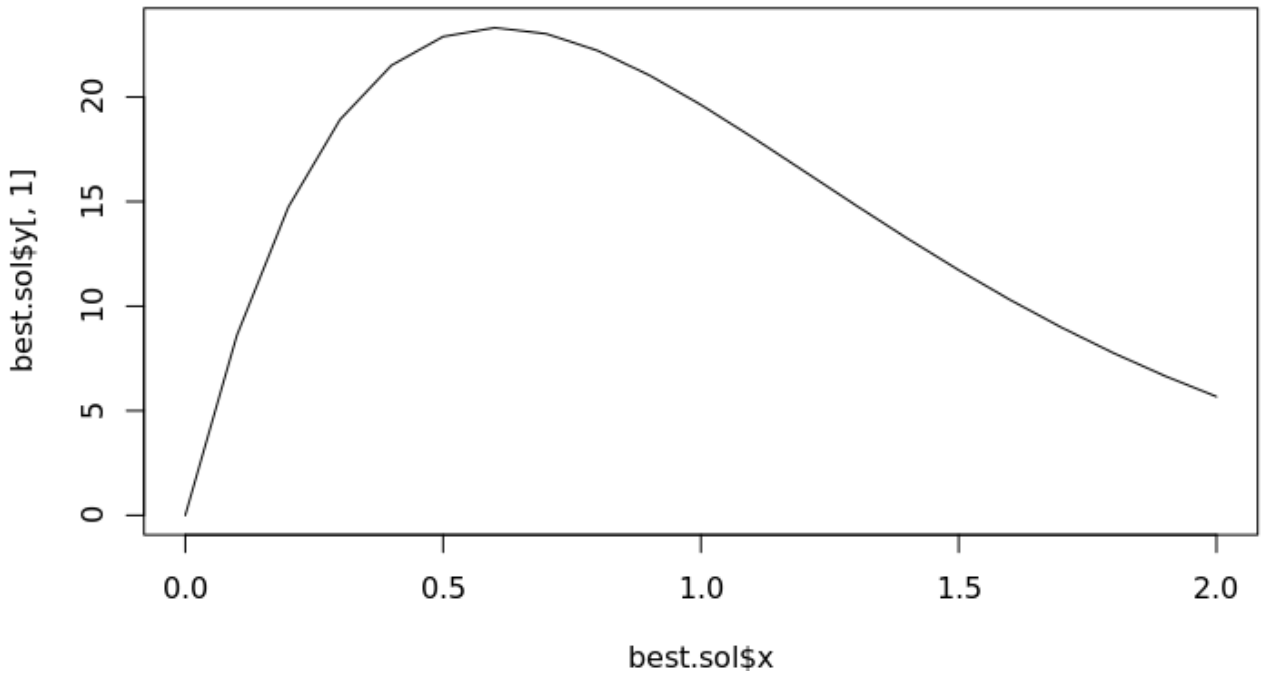
b.)

```
yo <- 0
ymax <- 0

proj.obj <- function(v0, y0=0, tfinal){
  # minimize w.r.t. v0
  proj.sol <- ode45(projectile.f,
                    y=c(y0, v0), t0=0, tfinal=tfinal)
  final_index <- length(proj.sol$t)
  yf <- proj.sol$y[final_index,1]  # want equal to right boundary
  log(abs(yf)) # minimize this
}

# user specifies tfinal and yfinal for BVP
v_best <- optimize(proj.obj,
                   interval=c(1,100), #bisect-esque interval
                   tol=1e-10,
                   y0=0, tfinal=10) # un-optimized obj params
v_best$minimum  # best v0

best.sol <- rk4sys(projectile.f, a=0, b=10, y0=c(0, v_best$minimum),
                   n=20)  # 20 integration stepstmax
plot(best.sol$x, best.sol$y[,1], type="l")

computeHeights <- function(xo, vo, tmin){
  height <- xo + vo*tmin -0.5*9.81*tmin^2
}

height <- computeHeights(yo, v_best$minimum, seq(0,10,len=21))
plot(seq(0,10,len=21), height, xlab="time", ylab="height")
```

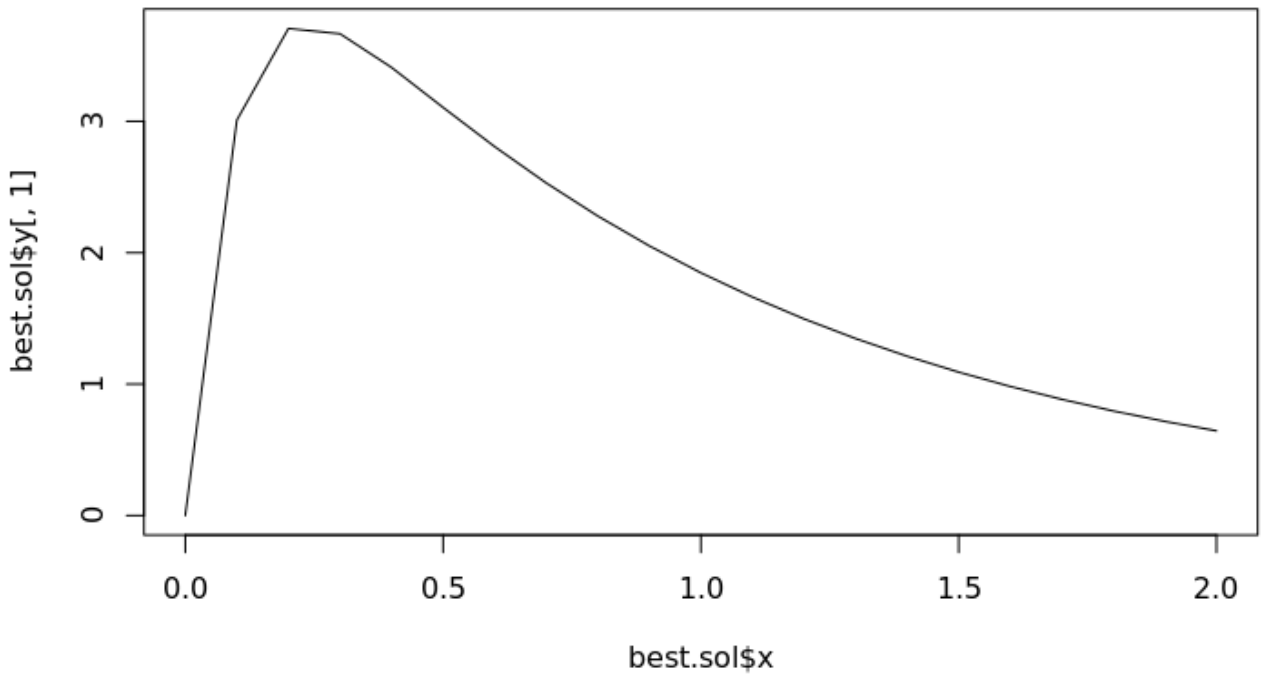


c.)

```
yo <- 0
y1 <- 1
tmin <- 0
tfinal <- 2

dho.pert.encap.f <- function(ep){
  dho.pert.f <- function(t, y){
    yn <- y[1]
    ydot <- y[2] - yn
    #ep <- 0.5
    ypp <- (-(1+ep)*yn - ydot)/ep
    as.matrix(c(ydot,ypp))
  }

  proj.obj <- function(v0, y0=yo, tfinal){
    # minimize w.r.t. v0
    proj.sol <- ode45(dho.pert.f,
                      y=c(yo, y1), t0=tmin, tfinal=tfinal)
    final_index <- length(proj.sol$t)
    yf <- proj.sol$y[final_index,1]  # want equal to right boundary
    log(abs(yf)) # minimize this
  }

  # user specifies tfinal and yfinal for BVP
  ydot_best <- optimize(proj.obj,
                        interval=c(1,100), #bisect-esque interval
                        tol=1e-10,
                        y0=0, tfinal=2) # un-optimized obj params
  ydot_best$minimum  # best ydot

  best.sol <- rk4sys(dho.pert.f, a=0, b=2, y0=c(0, ydot_best$minimum),
                     n=20)  # 20 integration stepstmax
}

ep <- 0.5
best.sol <- dho.pert.encap.f(ep)
plot(best.sol$x, best.sol$y[,1], type="l")
```




```
ep <- 0.05
best.sol <- dho.pert.encap.f(ep)
plot(best.sol$x, best.sol$y[,1], type="l")
```



What is the analytical solution when the perturbation parameter is 0?

yprime = -y
Same as e^x

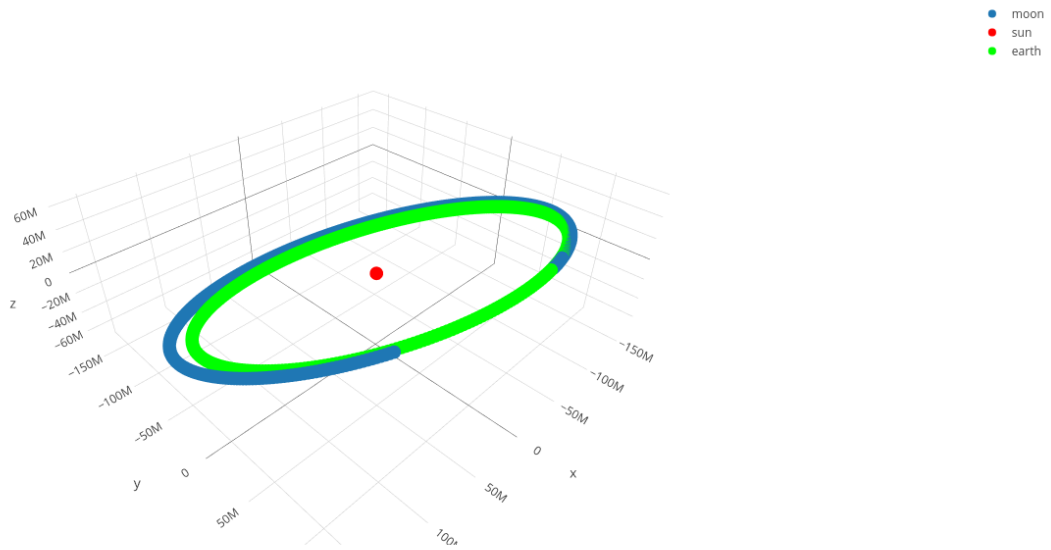
4.)

a.)

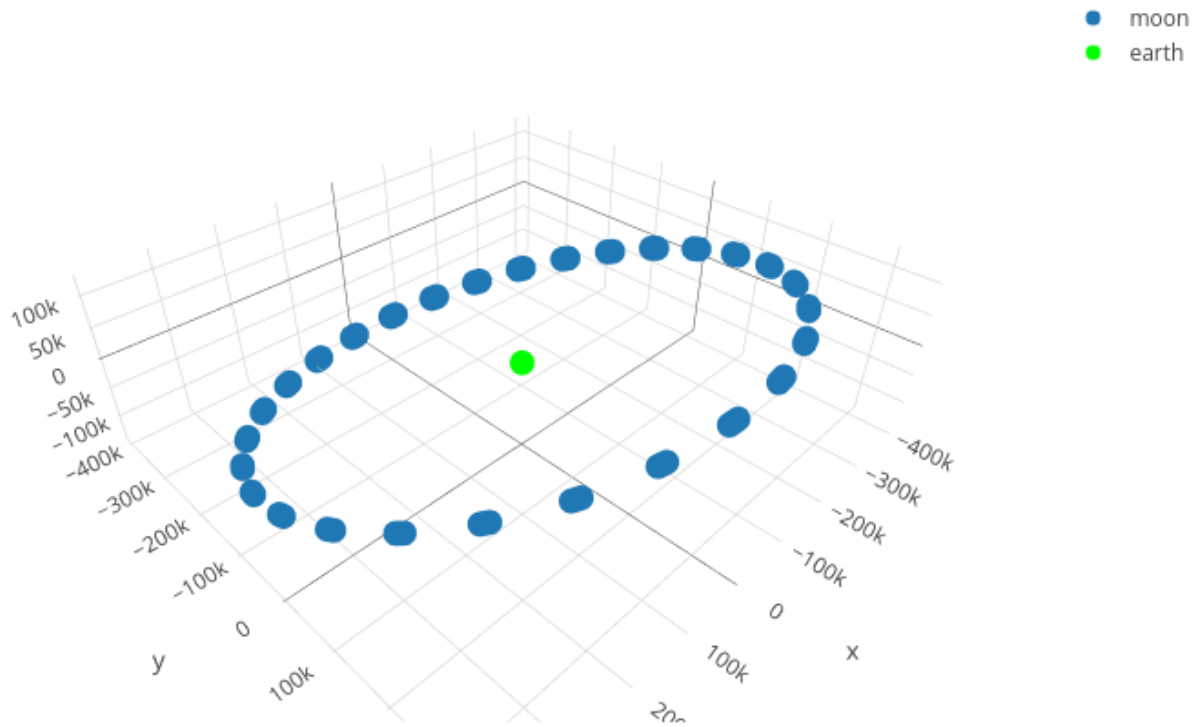
```
# t0: jan 1, 1999 00:00:00am
x0_earth <- c(-27115219762.4, 132888652547.0, 57651255508.0)/1e3 # km
v0_earth <- c(-29794.2199907, -4924.33333333,-2135.59540741)*(86400)/1e3
# m/s -> km/day

x0_moon <- c(-27083318944, 133232649728, 57770257344)/1e3 # km
v0_moon <- c(-30864.2207031, -4835.03349304, -2042.89546204)*(86400)/1e3
# m/s -> km/day

sunearth.f <- function(t,y){
  G <- 6.673e-11 # m^3 kg^-1 s^-2
  M_S <- 1.9891e30
  M_E <- 5.98e24
  M_m <- 7.32e22
  mu_sun <- G*M_S*(86400^2)/1e9  # km^3/days^2

  x_earth <- y[1:3]  # xyz position of earth wrt sun
  v_earth <- y[4:6]  # xyz velocity of earth, also part of ydot
  r_earth <- sqrt(sum(x_earth^2))  # distance earth to sun
  acc_earth <- -mu_sun*x_earth/r_earth^3  # part of ydot

  ydot <- c(v_earth, acc_earth)
  return(matrix(ydot))
}

y.init.earth <- c(x0_earth, v0_earth)
sunearth.sol <- rk4sys(f=sunearth.f, a=0, b=365.25, y0=y.init.earth, n=365)
sunearth.df <- data.frame(x=sunearth.sol$y[,1],
                          y=sunearth.sol$y[,2],
                          z=sunearth.sol$y[,3])

y.init.moon <- c(x0_moon, v0_moon)
sunmoon.sol <- rk4sys(f=sunearth.f, a=0,b=365.25, y0=y.init.moon, n=365)
sunmoon.df <- data.frame(x=sunmoon.sol$y[,1],
                          y=sunmoon.sol$y[,2],
                          z=sunmoon.sol$y[,3])

if (!require("plotly")) install.packages("plotly")
library(plotly)
fig <- plot_ly(sunmoon.df, x = ~x, y = ~y, z = ~z, name="moon", type = "scatter3d",
mode="markers")
fig <- fig %>% layout(scene = list(xaxis = list(title = 'x'),
                                   yaxis = list(title = 'y'),
                                   zaxis = list(title = 'z')))
fig <- add_trace(fig, x = 0, y = 0, z=0, mode="markers", color = I("red"),
name="sun")
fig <- add_trace(fig, x = sunearth.df[1]$x, y = sunearth.df[2]$y, z =
sunearth.df[3]$z, mode="markers", color = I("green"), name="earth")
fig

mean(sqrt(rowSums(cbind(sunearth.sol$y[,1]^2,
                        sunearth.sol$y[,2]^2,
                        sunearth.sol$y[,3]^2))))
# 150 million km
```

```
sunmoon.f <- function(t,y){
  G <- 6.673e-11 # m^3 kg^-1 s^-2
  M_E <- 5.98e24
  M_m <- 7.32e22
  mu_sun <- G*M_E*(86400^2)/1e9  # km^3/days^2

  x_moon <- y[1:3]  # xyz position of moon wrt earth
  v_moon <- y[4:6]  # xyz velocity of moon, also part of ydot
  r_moon <- sqrt(sum(x_moon^2))  # distance moon to earth
  acc_moon <- -mu_sun*x_moon/r_moon^3  # part of ydot

  ydot <- c(v_moon, acc_moon)
  return(matrix(ydot))
}

x0_moonearth <- x0_moon - x0_earth
v0_moonearth <- v0_moon - v0_earth
y.init.moonearth <- c(x0_moonearth, v0_moonearth)
earthmoon.sol <- rk4sys(f=sunmoon.f, a=0,b=365.25, y0=y.init.moonearth, n=365)
earthmoon.df <- data.frame(x=earthmoon.sol$y[,1],
                           y=earthmoon.sol$y[,2],
                           z=earthmoon.sol$y[,3])

mean(sqrt(rowSums(cbind(earthmoon.sol$y[,1]^2,
                        earthmoon.sol$y[,2]^2,
                        earthmoon.sol$y[,3]^2))))

fig <- plot_ly(earthmoon.df, x = ~x, y = ~y, z = ~z, name="moon", type =
"scatter3d", mode="markers")
fig <- fig %>% layout(scene = list(xaxis = list(title = 'x'),
                                   yaxis = list(title = 'y'),
                                   zaxis = list(title = 'z')))
fig <- add_trace(fig, x = 0, y = 0, z = 0, mode="markers", color = I("green"),
name="earth")
fig
```

Note: This first Figure does NOT account for moon's rotation around the sun. This is just an overlay of the Earth's rotation around the Sun and the Moon's rotation around the sun.



This second Figure accounts for the moon's rotation around the Earth as it is static in space over time.



What is the average distance between earth and sun over a year?
150,039,214 km

What is the average distance between earth and moon?
390,598.8 km

b.)

```
norm_vec <- function(x) sqrt(rowSums(cbind((x^2))))
eclipses <- norm_vec(sunearth.sol$y) %*% norm_vec(sunmoon.sol$y)
```

```
          [,1]
[1,] 9.074151e+18
```

c.)

```
# c. keep orbiter at L2 Lagrange point for a year, ignoring the moon's effect
findZeroRelax <- function(g, x.guess, tol=1e-6, maxsteps=1e6){
  steps <- 0
  x.old <- x.guess
  isConverged <- FALSE
  while (!isConverged){
    x.new <- g(x.old)
    if(steps >= maxsteps || (abs(x.new-x.old)<tol)){
      isConverged <- TRUE
    }
    else{
      x.old <- x.new
    }
    steps <- steps + 1
  }
  return(c(x.new, g(x.new), steps))
}

T <- 365.25
G <- 6.673e-11 # m^3 kg^-1 s^-2
M_S <- 1.9891e30
M_E <- 5.98e24
x_earth <- c(-27115219762.4, 132888652547.0, 57651255508.0)/1e3 # km
r_earth <- sqrt(sum(x_earth^2))  # distance earth to sun

orbiter.f <- function(l) {
  tmp1 <- ((T^2)/(4*(pi^2))) * G
  tmp2 <- (M_S/((r_earth+l)^2)) + (M_E/(l^2))
  tmp3 <- tmp1*tmp2
}

orbiter.estimate <- findZeroRelax(orbiter.f, 1000000)
l <- orbiter.estimate[1]-r_earth
```

```
x0_orbiter_tmp <- c(l, orbiter.f(l), 0)/1e3 # km
x0_orbiter <- x0_orbiter_tmp + x0_earth

y.init.orbiter <- c(x0_orbiter, v0_earth)
sunorbit.sol <- rk4sys(f=sunearth.f, a=0, b=365.25, y0=y.init.orbiter, n=365)

sunorbit.df <- data.frame(x=sunorbit.sol$y[,1],
                          y=sunorbit.sol$y[,2],
                          z=sunorbit.sol$y[,3])

fig <- plot_ly(sunorbit.df, x = ~x, y = ~y, z = ~z, name="orbiter", type =
"scatter3d", mode="markers")
fig <- fig %>% layout(scene = list(xaxis = list(title = 'x'),
                                   yaxis = list(title = 'y'),
                                   zaxis = list(title = 'z')))
fig <- add_trace(fig, x = 0, y = 0, z=0, mode="markers", color = I("red"),
name="sun")
fig <- add_trace(fig, x = sunearth.df[1]$x, y = sunearth.df[2]$y, z =
sunearth.df[3]$z, mode="markers", color = I("green"), name="earth")
fig
```