Physics 2211 - Lab 3

Black Hole

Advaith Menon[1]

[1]Computer Engineering
Georgia Institute of Technology

February 29th, 2024

Outline

Outline

Aim

Purpose of this lab assignment

- Given the mass and trajectory of a star that orbits around a black hole, find the mass of the back hole.
- Find the velocity of a spaceship of a given mass, which is on the planet, such that:
  - The spaceship just passes the black hole
  - The spaceship orbits around the black hole

Outline

Newton's Second Law

Quantitative analysis of motion

"The net force acting on a body is defined as the change in its momentum per unit time."

$$\vec{F}_{net} = \frac{\mathrm{d}\vec{p}}{\mathrm{d}t} \tag{1}$$

In most daily life scenarios, mass doesn't change with respect to time, hence this equation is better known as:

$$\vec{F}_{net} = m \cdot \vec{a}$$

where,

- $\vec{F}_{net} =$ Net force acting on a body
- $m =$ Mass of the body
- $\vec{a} = \frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} =$ Net acceleration on body
- $\Delta\vec{p} = m \times \Delta\vec{v} =$ Change in momentum

Newton's Universal law of Gravitation

"The gravitational force acting of an object is directly proportional to the mass of the two objects, and inversely proportional to the square of the distance between the planets."

$$\vec{F}_{g,21} = \frac{Gm_1m_2}{|\vec{r}_{21}|^2} \cdot \hat{r}_{12} \tag{2}$$

where,

- $\vec{F}_{g,21} =$ The gravitational force on object 1 due to object 2.
- $m_1 =$ Mass of the first body
- $m_2 =$ Mass of the second body
- $\vec{r}_{21} = \vec{r}_2 - \vec{r}_1 =$ The vector "distance" between the planets Change in momentum

Initial Conditions - Part 1

- Position of the black hole, $\vec{r}_{BH} = \langle 0, 0, 0 \rangle$ m
- Position of the star, $\vec{r}_{star}$ given to us in a Python List at given time intervals.
- Mass of the star, $m_{star} = 2 \times 10^{30}$ kg
- $\Delta t = 86400$ s

Initial Conditions - Part 2

- Position of the black hole, $\vec{r}_{BH} = \langle 0, 0, 0 \rangle$ m
- Mass of the blackhole, $m_{BH} = 1.35641 \times 10^{34}$ kg (from part 1)
- Initial position of the planet, $\vec{r}_{planet} = \langle 1 \times 10^{10}, 0, 0 \rangle$ m
- Radius of the planet, $a = 1 \times 10^{8}$ m
- Initial position of the ship, $\vec{r}_{ship} = \langle 1 \times 10^{10}, 1 \times 10^{8}, 0 \rangle$ m

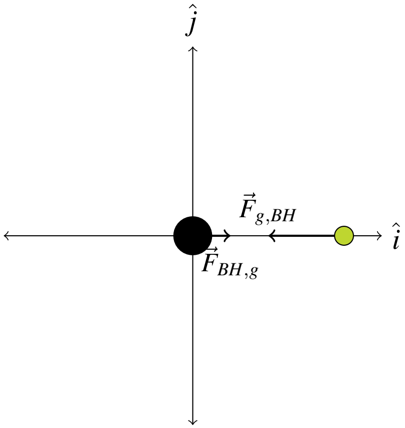
System and Surroundings - Part 1

- **System:** The star (hereafter referred to ‘object’)
- **Surroundings:** Everything else (black hole etc.)

System and Surroundings - Part 2

- **System:** The ship (hereafter referred to ‘object’)
- **Surroundings:** Everything else (black hole, planet etc.)

Diagram



Outline

Formulae - Experiment 1

Resolving the net forces on each object

- $\vec{p}_i = m_{star} \cdot \vec{v}_{init}, \vec{p}_i = m_{star} \cdot \vec{v}_{final}$
- $\Delta\vec{v} = \vec{v}_{final} - \vec{v}_{init}, \Delta\vec{p} = \vec{p}_{final} - \vec{p}_{init}$
- $\vec{F}_{net} = \dfrac{\Delta\vec{p}}{\Delta t}$
- $\left(\dfrac{\mathrm{d}\vec{p}}{\mathrm{d}t}\right)_{\parallel} = |\vec{F}_{net}||\vec{p}|\cos(\theta) = \vec{F}_{net} \cdot \hat{p} \times \hat{p}$
- $\left(\dfrac{\mathrm{d}\vec{p}}{\mathrm{d}t}\right)_{\perp} = \vec{F}_{net} - \left(\dfrac{\mathrm{d}\vec{p}}{\mathrm{d}t}\right)_{\parallel}$

Formulae - Experiment 2

Resolving the net forces on each object

$$\vec{F}_{g,21} = \frac{Gm_1m_2}{|\vec{r}_{21}|^2} \cdot \hat{r}_{12} \tag{3}$$

Simulation code

Finding mass of black hole

```
seed="amenon301@gatech.edu"

# ... kepler's formula

scene.background = color.white

star = sphere(color=color.yellow,
↪   radius=3e11)
trail = curve(color=color.green)
origin = sphere(pos=vector(0,0,0),
↪   color=color.black, radius=3e11)

arrowFnet = arrow(pos=star.pos,
↪   axis=vector(0,0,0),
↪   color=color.orange)
arrowFnet_par = arrow(pos=star.pos,
↪   axis=vector(0,0,0),
↪   color=color.magenta)
arrowFnet_perp = arrow(pos=star.pos,
↪   axis=vector(0,0,0), color=color.cyan)

G = 6.7e-11

# Mass of the star -- EDIT THIS LINE
star.m = 2e30

# Time interval between imported data
↪   points -- EDIT THIS LINE
```

```
t = t + deltat

idx=1

myballs = 0

while idx <(len(X)-1):
    rate(50)

    v_init =
    ↪   vector(Xvel[idx-1],Yvel[idx-1],0)
    v_final =
    ↪   vector(Xvel[idx],Yvel[idx],0)

    # EDIT THESE TWO LINES
    p_init = star.m * v_init
    p_final = star.m * v_final # $\vec{p}
    ↪   = m\vec{v}$

    # EDIT THESE TWO LINES
    deltav = v_final - v_init
    deltap = p_final - p_init

    # EDIT THIS LINE
    dpdt = deltap/deltat

    # EDIT THIS LINE
    Fnet = dpdt
```

Georgia Tech

Simulation code

Finding mass of black hole

```
# EDIT THIS LINE
    Fnet_perp = dpdt_perp # yes.

    # Calculate the mass of the black
    ↪   hole
    # EDIT AND ADD LINES AS NEEDED
    #
    # or, F = (G*star.m*mBH)/(r^2)
    # mBH = (F * r^2)/(G * star.m)

    mBH = (mag(Fnet) * (X[idx]**2 +
    ↪   Y[idx]**2))/(G * star.m)
    myballs += mBH
    print("Mass of Black Hole", mBH)
    # print("Velocity of ball", v_init)

    # Update current position and
    ↪   velocity and show the object's
    ↪   current track
    star.pos = vector(X[idx],Y[idx],0)
    trail.append(pos=star.pos)

    # EDIT THE NEXT SEVEN LINES
    arrowscale = 1E-17      # determines
    ↪   how long to draw the arrows that
    ↪   represent vectors
    arrowFnet.pos = star.pos
    arrowFnet.axis = Fnet*arrowscale
    arrowFnet_par.pos = star.pos
    arrowFnet_par.axis =
    ↪   Fnet_par*arrowscale
    arrowFnet_perp.pos = star.pos
    arrowFnet_perp.axis =
    ↪   Fnet_perp*arrowscale

    t = t + deltat
    idx = idx + 1

print("Average mass of black hole",
↪   myballs/(idx-1))
print("|Fnet|=",mag(Fnet))
print("|Fnet_par|=",mag(Fnet_par))
print("|Fnet_perp|=",mag(Fnet_perp))
print("All done!")
```

Simulation code

Finding trajectory of the ship

```
Glowscript 3.2 VPython
## PHYS 2211
## Lab 3: Black Hole - Part 2
## 2211-lab3workingPART2.py
## Last updated: 2022-05-14 EAM



## ======================================
## VISUALIZATION & GRAPH INITIALIZATION
## ======================================

# White background so the black hole is
↪  visible
scene.background = color.white

# Visualization (object, trail, origin)
bh = sphere(pos=vec(0,0,0),
↪  color=color.black, radius=2e8)
planet = sphere(pos=vec(1e10,0,0),
↪  color=color.cyan, radius=1e8)
trailplanet = curve(color=color.cyan)
ship =
↪  sphere(pos=vec(planet.pos.x,planet.radius),
↪  color=color.magenta, radius=3e7)
trailship = curve(color=color.magenta)
```

```
##
↪  ========================================
## SYSTEM PROPERTIES & INITIAL CONDITIONS
##
↪  ========================================


# Mass of the black hole
# EDIT THIS LINE with your result from
↪  Part 1 of the lab
bh.m = 1.35641e+34


# Other constants - DO NOT CHANGE THESE
↪  FIVE LINES
G = 6.7e-11             # gravitational
↪  constant
planet.m = 9e28         # mass of planet
ship.m = 7e10           # mass of
↪  spaceship
deltat = 1              # deltat (in
↪  seconds)
t=0                     # initial time
# DO NOT CHANGE THE ABOVE FIVE LINES

def f_g(m_1, m_2, r, debug=False):
    rmag = r
    if isinstance(r, vec):
```

Georgia Tech

Simulation code

Finding trajectory of the ship

```
# Maximum time to run the simulation
# Default is ONE HOUR, but you can change
↪   it to a longer
# amount (e.g., 2hrs, 5hrs, etc) if you
↪   need more time to
# visualize a full orbit
tmax=60*60*3


# Initial conditions for the planet
#planet_speed = sqrt(G * bh.m /
↪   mag(planet.pos) )
#planet.vel = vec(0,planet_speed,0)
#print("Planet's orbital speed:",
↪   planet_speed, "m/s")
#print("Planet's orbital period:",
↪   (2*pi*mag(planet.pos)/planet_speed)/(60*60),
↪   "hrs")
#print("---")
#
#
## Initial conditions for the spaceship
## EDIT THESE LINES as needed to change
↪   the initial velocity of the ship
## Remember that your spaceship cannot
↪   move faster than the speed of light
#ship_speed_x = 0
#ship_speed_y = 0
```

```
def resetConds():
    bh.pos = vec(0,0,0)
    planet.pos = vec(1e10,0,0)
    trailplanet.clear()
    ship.pos =
    ↪   vec(planet.pos.x,planet.radius,0)
    trailship.clear()
    planet_speed = sqrt(G * bh.m /
    ↪   mag(planet.pos) )
    planet.vel = vec(0,planet_speed,0)
    print("Planet's orbital speed:",
    ↪   planet_speed, "m/s")
    print("Planet's orbital period:",
    ↪   (2*pi*mag(planet.pos)/planet_speed)/(60*6
    ↪   "hrs")
    print("---")

    # Initial conditions for the
    ↪   spaceship
    # EDIT THESE LINES as needed to
    ↪   change the initial velocity of
    ↪   the ship
    # Remember that your spaceship cannot
    ↪   move faster than the speed of
    ↪   light
#   ship_speed_x = 0
#   ship_speed_y = 0
```

Simulation code

Finding trajectory of the ship

```
# Maximum time to run the simulation
# Default is ONE HOUR, but you can change
↪  it to a longer
# amount (e.g., 2hrs, 5hrs, etc) if you
↪  need more time to
# visualize a full orbit
tmax=60*60*3


# Initial conditions for the planet
#planet_speed = sqrt(G * bh.m /
↪  mag(planet.pos) )
#planet.vel = vec(0,planet_speed,0)
#print("Planet's orbital speed:",
↪  planet_speed, "m/s")
#print("Planet's orbital period:",
↪  (2*pi*mag(planet.pos)/planet_speed)/(60*60),
↪  "hrs")
#print("---")
#
#
## Initial conditions for the spaceship
## EDIT THESE LINES as needed to change
↪  the initial velocity of the ship
## Remember that your spaceship cannot
↪  move faster than the speed of light
#ship_speed_x = 0
#ship_speed_y = 0
```

```
def resetConds():
    bh.pos = vec(0,0,0)
    planet.pos = vec(1e10,0,0)
    trailplanet.clear()
    ship.pos =
    ↪  vec(planet.pos.x,planet.radius,0)
    trailship.clear()
    planet_speed = sqrt(G * bh.m /
    ↪  mag(planet.pos) )
    planet.vel = vec(0,planet_speed,0)
    print("Planet's orbital speed:",
    ↪  planet_speed, "m/s")
    print("Planet's orbital period:",
    ↪  (2*pi*mag(planet.pos)/planet_speed)/(60*6
    ↪  "hrs")
    print("---")

    # Initial conditions for the
    ↪  spaceship
    # EDIT THESE LINES as needed to
    ↪  change the initial velocity of
    ↪  the ship
    # Remember that your spaceship cannot
    ↪  move faster than the speed of
    ↪  light
#   ship_speed_x = 0
#   ship_speed_y = 0
```

Simulation code

Finding trajectory of the ship

```
## ================
## CALCULATION LOOP
## ================
def calcLoop():
    t = 0
    while t < tmax:
        rate(1000)

        # Net force on the planet
        # EDIT THESE LINES (and/or add
        ↪  more if needed) to find:
        # 1) Fgrav on the planet by the
        ↪  black hole
        # 2) Fgrav on the planet by the
        ↪  spaceship
        # 3) the net force on the planet
        r_bh_to_planet = planet.pos
        F_bh_ON_planet = f_g(planet.m,
        ↪  bh.m, -r_bh_to_planet)

        r_ship_to_planet =
        ↪  planet.pos-ship.pos
        F_ship_ON_planet = f_g(ship.m,
        ↪  planet.m, -r_ship_to_planet)

        Fnet_ON_planet = F_bh_ON_planet +
        ↪  F_ship_ON_planet
```

```
# 2) Fgrav on the spaceship by
↪  the planet
# 3) the net force on the
↪  spaceship
r_bh_to_ship = ship.pos
F_bh_ON_ship = f_g(ship.m, bh.m,
↪  -r_bh_to_ship)

r_planet_to_ship =
↪  ship.pos-planet.pos
F_planet_ON_ship = f_g(ship.m,
↪  planet.m, -r_planet_to_ship)

Fnet_ON_ship = F_bh_ON_ship +
↪  F_planet_ON_ship
# print(Fnet_ON_ship)


# MAKING THINGS MOVE
# Apply Newton's 2nd law and the
↪  position update formula
# to make the planet and
↪  spaceship move
planet.vel = planet.vel +
↪  (Fnet_ON_planet/ planet.m) *
↪  deltat
planet.pos = planet.pos +
↪  planet.vel * deltat
```

Simulation code

Finding trajectory of the ship

```python
        # Adding break conditions, in case of a crash
        # Do not edit these lines
        if mag(r_bh_to_ship) < bh.radius:
            # print("Oh no, you got sucked into the black hole!")
            # break
            return False
        if mag(r_ship_to_planet) < planet.radius:
            # print("Oh no, you crashed into the planet!")
            # break
            return False

        trailplanet.append(pos=planet.pos) # Planet is cyan
        trailship.append(pos=ship.pos) # Spaceship is magenta

        t = t+deltat
    return True;

def bruteForceIsHowNasaLaunchesSatellites():
```

```python
# for i in range(-int(3e8), int(3e8), int(1e5)):
    # for j in range(-int(3e7), int(3e7), int(1e6)):
        i = -1e8 # -1e1 # -1e8
        j = 1e7 # 1e7 # 1e7
        # reset conds
        resetConds()
        vel = vec(i, j, 0)
        ship.vel = vec(i, j, 0)
        print("Try! v =", vel)
        if calcLoop():
            print("This works! v =", vel)

bruteForceIsHowNasaLaunchesSatellites()
#print("---")
#print("Calculations finished after ",t/60,"minutes")
#print("The ship is now", mag(ship.pos)-bh.radius, "m away from the event horizon of the black hole")
#print("All done!")
```

Results

- Mass of the black hole, $m_{BH} = 1.35641 \cdot 10^{34}$ kg
- Velocity to visit the black hole, $\vec{v}_{ship} = \langle -1 \times 10^8, 1 \times 10^7, 0 \rangle$ ms^{-1}
- Velocity to orbit the black hole, $\vec{v}_{ship} = \langle -1 \times 10^1, 1 \times 10^7, 0 \rangle$ ms^{-1}

Outline

What does it mean?

Parallel and perpendicular components

- From Newton's Second Law, we now that the net force is the change in net momentum per unit time.
- However, $\frac{\Delta\vec{p}}{\Delta t} = \frac{\Delta m \times \Delta\vec{v}}{\Delta t}$, which means that we need mass, velocity and time taken to calculate rate of change of momentum.
- Force is an abstraction that can be used in other contexts, for example, the Work-Energy Theorem:

$$\Delta K = W(= \vec{F} \cdot \Delta s)$$

What if...

...we wanted to orbit the black hole?

- Sure thing! By using logical brute-forcing, I got this answer:

$$\vec{v}_{ship} = \langle -1 \times 10^1, 1 \times 10^7, 0 \rangle \text{ ms}^{-1}$$