Modeller och verklighet för datateknik VT 2024

Arbetsuppgift 1:
Satellitdockning

Grupp: 2
Namn: Firoz-Khan Akbari
Namn: Bilal Ayubi
Namn: Ömer Kolsuz
Namn: Oscar Svantesson
Datum för inlämning: 2024-01-18

1 Inledning

Vi ska simulera en situation där en satellit dockar i en annan satellit som befinner sig i vila. Detta innebär att vi ska hantera de fysiska aspekterna av dockningen i vår simuleringskod. Yttre krafter som gravitation och luftmotstånd kan försummas samt att satelliterna rör sig längs räta linjer istället för i omloppsbanor. Den satellit som är i rörelse kan röra sig framåt och bakåt genom raketkrafter. Koden skall vara uppbyggd på följande: Det ska skrivas i en funktion som tar in information om satelliternas position, hastighet, kraften som påverkar den rörliga satelliten, och storleken på tidssteget. Funktionen ska beräkna satelliternas nya positioner och hastigheter för nästa tidssteg. Funktionen ska även avgöra om dockning kan ske inom detta tidssteg och hantera rörelsen efter dockningen beroende på om det är en inelastisk stöt eller elastisk kollision.

2 Teoretisk bakgrund

Begreppet hastighet är ett begrepp inom fysiken som enligt (fysik.org, Läge, Hastighet, Acceleration) säger hur snabbt någonting förflyttar sig samt vilken riktning. Formeln för att beräkna en förflyttning under tidsintervallet Δt ges genom:

$$Förflyttning \;=\; Hastighet\,(v) \;\times\; tidsintervallet(\Delta t) \;=\; m/s$$

Begreppet Acceleration är ett begrepp inom fysiken som enligt (fysik.org, Läge, Hastighet, Acceleration) förklarar en förändring i hastigheten, detta beräknas genom följande:

$$Acceleration\,(a) \;=\; \frac{\Delta v}{\Delta t} \;=\; m/s^2$$

Tidsdiskret simulering bryter tiden i steg och beräknar hastighet och acceleration vid fasta tidpunkter. Tidskontinuerlig simulering behandlar tiden som en kontinuerlig variabel och ger mer exakt beskrivning av systemets beteende genom kontinuerliga funktioner.
Inom fysik kan ett friläggningsdiagram användas för att visualisera hur energi eller krafter flödar och omvandlas inom ett fysiskt system. I vårt fall kan vi koppla det till potentiell energi och kinetisk energi.

Newtons andra lag har betydelse vid behandlingen av problem t.ex om vi vet vilken kraft som tillämpas på ett föremål och vad föremålets massa är, kan vi räkna ut hur snabbt föremålet accelererar och utifrån detta kan vi då få fram vilken hastighet som föremålet har.

$$F \;=\; m \;\times\; a \;=\; m/s^2$$

Newtons tredje lag har betydelse vid behandlingen av problem t.ex om objekt A påverkar objekt B med en kraft, så kommer objekt B att påverka objekt A med en kraft av samma storlek och i motsatt riktning. I uppgiften så är satelliten rödmarkerad och föremålet som skall kollidera med satelliten vitmarkerad. Kör man ett test där hastigheten är låg så sker inte så stor kraft bakåt, men ökar vi kraften så ökas hastigheten och när kollisionen sker så ser man hur föremålet som kolliderar med satelliten flyger bakåt med en kraft, lite som en studsboll tillbaka samma håll som de kom ifrån.

När ett system är i mekanisk jämvikt betyder det att det inte upplever någon acceleration i någon riktning. Det innebär att summan av alla externa krafter som verkar på systemet är noll, och summan av alla externa vridande krafter om någon punkt är också noll.

Tröghet är en egenskap hos ett objekt som beskriver dess motvilja att ändra sin rörelsestatus. Det betyder att ett objekt i vila tenderar att förbli i vila, medan ett objekt i rörelse har en tendens att fortsätta röra sig med konstant hastighet i samma riktning, om det inte påverkas av yttre krafter.

Rörelsemängd är hur mycket rörelse det finns i ett objekt och detta beräknas genom o ta objektets massa massa och dess hastighet:
$p = m \times v = m/s$

Impuls beskriver hur snabbt rörelsemängd förändras, detta beräknas genom:
$I = N \times s = Ns$

En elastisk stöt och en oelastisk stöt är två olika typer av kollisioner mellan objekt, och de karakteriseras av hur mycket kinetisk energi som bevaras efter kollisionen. I en elastisk stöt bevaras den totala kinetiska energin före och efter kollisionen. Det innebär att ingen kinetisk energi förloras under kollisionen. I en oelastisk stöt förloras en del av den totala kinetiska energin under kollisionen. Efter kollisionen kommer objekten att röra sig tillsammans som en enhet, och de kommer att vara i kontakt med varandra efter kollisionen.

För att rörelsemängden ska bevaras måste tre villkor uppfyllas, dessa är följande:
Avsaknad av Yttre Krafter: För att rörelsemängden i ett system ska vara konstant och därmed bevaras, måste det inte finnas några yttre krafter som verkar på systemet. Detta innebär att det inte får vara några nettokrafter som påverkar objekten i systemet från omgivningen.
Lag om Växling (Lagomväxlingsprincipen): Principen om rörelsemängdens bevarande innebär att för varje kraft som verkar på ett objekt i systemet måste det finnas ett annat objekt i systemet som upplever en lika stor kraft men i motsatt riktning. Detta säkerställer att rörelsemängden i systemet förblir konstant.
Stängt System: För att tillämpa rörelsemängdens bevarande måste systemet betraktas som stängt. Det innebär att inga objekt kan lämna eller läggas till i systemet under den aktuella tidsperioden eller det specifika händelseförlopp som studeras. Detta är nödvändigt för att säkerställa att den totala rörelsemängden i systemet förblir oförändrad.

Rörelseenergi är den energi som ett objekt har på grund av dess rörelse. Den beror på objektets massa och dess hastighet. Rörelseenergi får vi genom att ta:

$E = \frac{1}{2} \times mv^2 = Joule$

Lägesenergi är den energi som är förknippad med objektets position eller höjd över en punkt, vanligtvis jordens yta. Det är energin som kan frigöras eller omvandlas när objektet rör sig i förhållande till sitt läge. Lägesenergi beräknas med:

$p = mgh = Joule$
Mekanisk energi, som består av både kinetisk energi (rörelseenergi) och potentiell energi (lägesenergi), bevaras om det inte verkar några externa krafter på systemet.

Elektrisk energi, måste elektriska kretsar vara stängda kretsar där ingen ström förloras till ledningsmotstånd eller andra förluster.

Termisk energi, för att bevara termisk energi måste det inte finnas någon värmeöverföring till eller från systemet. Isolerade system, där värmeöverföring är noll, bevarar termisk energi.

Kärnenergi, för att bevara kärnenergi, måste vara stabila atomkärnor eller en situation där inga kärnreaktioner äger rum.

Kemisk energi, för att bevara kemisk energi, måste inga kemiska reaktioner äga rum. I en sluten och isolerad kemisk reaktion kommer den totala kemiska energin att bevaras.

Masscentrum, rörelse och total rörelsemängd är alla kopplade och kan beskrivas med hjälp av två punkter:

1. Den totala rörelsemängden av ett isolerat system förblir konstant om inga yttre krafter verkar på systemet.
2. När inga yttre krafter verkar på ett isolerat system, kommer systemets totala rörelse (hastighet och riktning) att förbli konstant. Detta innebär att systemets masscentrum kommer att röra sig med konstant hastighet om inga yttre krafter påverkar systemet.

Newtons andra lag beskriver sambandet mellan en kraft som verkar på ett föremål, föremålets massa, och den acceleration som kraften orsakar. Lagen formuleras som $F = m \bullet a$, där "F" är kraften som verkar på föremålet, "m" är föremålets massa och "a" är accelerationen som föremålet får. Om vi vet vilken kraft som tillämpas på ett föremål och vad föremålets massa är, kan vi räkna ut hur snabbt föremålet accelererar.[3]

Likformigt föränderlig rörelse formeln beskriver sambandet mellan sträckan ett föremål rör sig, dess starthastighet, acceleration, och den tid rörelsen pågår. Formeln ser ut så här: $s = s_0 + v_0 \bullet t + \frac{a \bullet t^2}{2}$, där "s" är föremålets position vid tiden "t", "s_0" är föremålets start position, "v_0"" är föremålets starthastighet och "a" är accelerationen.[10] När hastigheten inte förändras är accelerationen 0 och då kan man få sträckan genom att förenkla formeln till följande, $s = s_0 + v_0 \bullet t$. När accelerationen är konstant och man vill beräkna den nya hastigheten då kan man förenkla formeln till följande, $v = v_0 + a \bullet t$.[11]

I en fullständigt inelastisk stöt kolliderar två föremål och fastnar ihop, sedan rör de sig som ett enda föremål efter kollisionen. Den totala rörelsemängden före stöten är lika med den totala rörelsemängden efter stöten, $p_{före} = p_{efter}$, $p = m \bullet v$, där m är massan för föremålet och v är föremålets hastighet. Om vi har två föremål som kolliderar med fullständigt

inelastisk stöt kan vi utveckla formeln till, $m_1 \bullet v_1 + m_2 \bullet v_2 = (m_1 + m_2) \bullet v_{efter}$. Vill man beräkna hastigheten efter stöten kan man bryta ut "v_{efter}" vilket ger,

$v_{efter} = \frac{m_1 \bullet v_1 + m_2 \bullet v_2}{m_1 + m_2}$. När det är elastisk stöt är inte bara rörelsemängden bevarande utan även rörelseenergin, det vill säga att, $E_{före} = E_{efter}$, där $E = \frac{m \bullet v^2}{2}$. När föremålen kolliderar i en elastisk stöt fastnar de inte ihop som de gör i en fullständigt inelastisk stöt. För att få hastigheten för ett av föremålen efter stöten utesluter vi variabeln.

$m_1 \bullet v_{före\,1} + m_2 \bullet v_{före\,2} = m_1 \bullet v_{efter\,1} + m_2 \bullet v_{efter\,2} \Leftrightarrow v_{efter\,2} = \frac{m_1 \bullet v_{före\,1} + m_2 \bullet v_{före\,2} - m_1 \bullet v_{efter\,1}}{m_2}$.

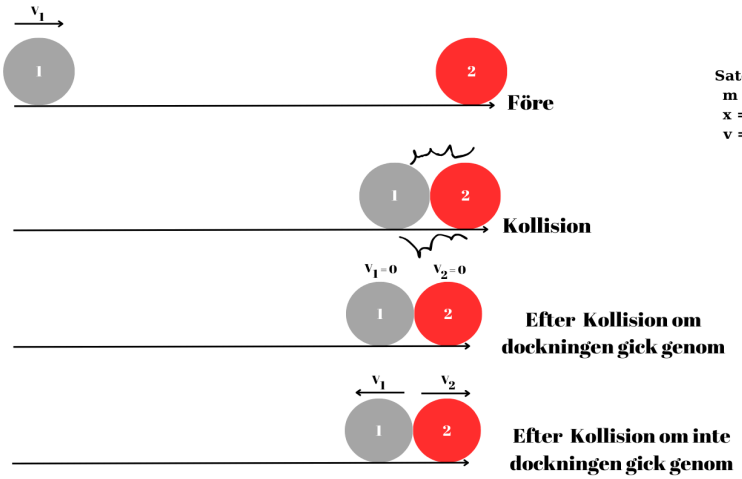
Den ekvationen vi nu har fått för hastigheten efter stöten kan vi nu sätta in i ekvationen för rörelseenergin, $E_{före} = E_{efter} \Rightarrow \frac{m_1 \bullet v_{före\,1}^2}{2} + \frac{m_2 \bullet v_{före\,2}^2}{2} = \frac{m_1 \bullet v_{efter\,1}^2}{2} + \frac{m_2 \bullet v_{efter\,2}^2}{2} \Rightarrow$

$\frac{m_1 \bullet v_{före\,1}^2}{2} + \frac{m_2 \bullet v_{före\,2}^2}{2} = \frac{m_1 \bullet v_{efter\,1}^2}{2} + \frac{m_2 \bullet \left(\frac{m_1 \bullet v_{före\,1} + m_2 \bullet v_{före\,2} - m_1 \bullet v_{efter\,1}}{m_2}\right)^2}{2}$, Om vi algebraiskt förenklar uttrycket kan vi få ekvationer för hastigheterna efter stöten,

$v_{efter\,1} = \frac{(m_1 - m_2) \bullet v_{före\,1} + 2 \bullet m_2 \bullet v_{före\,2}}{m_1 + m_2}$ & $v_{efter\,2} = \frac{(m_2 - m_1) \bullet v_{före\,2} + 2 \bullet m_1 \bullet v_{före\,1}}{m_1 + m_2}$. [9]

3 Utförande

Vi började med att försöka rita ut scenariot på papper och göra olika modelleringar om vilka fysik lagar som kan vara relevanta för de parametrarna som vi har, sträcka, hastighet, massa, kraft och tid.



När man ökar raketkraften ska raketens hastighet även ökas. För att kunna få den hastigheten behöver vi ta reda på accelerationen och det gör vi genom att använda oss av Newtons andra lag [1]. Nu kan vi bestämma raketens nya hastigheten med hjälp av likformigt föränderlig rörelse formeln [2]. Vi behöver även bestämma satelliternas positioner i x-led, det gör vi genom att använda oss av likformigt föränderlig rörelse formeln igen. I och med att vi tar hänsyn till accelerationen i [2] genom den nya hastigheten, räknas positionerna ut med endast

tidigare hastighet, acceleration och tid. Efter det la vi till en yttre if-sats för att kolla om avståndet mellan de två satelliterna är mindre än fem meter [3] och två if-satser inne i den yttre som avgör om dockningen lyckas eller misslyckas baserat på den kommande satellitens hastighet [4]. I den if-sats som hanterar vad som ska ske om dockningen lyckas sker det en fullständig inelastisk stöt och hastigheten efter stöten får vi genom att sätta in parametrarna i det uttrycket som uteslöt hastigheten efter stöten för en fullständig inelastisk stöt [5]. I den if-sats som hanterar vad som ska ske när det sker en elastisk stöt behöver vi beräkna en ny hastighet och en ny position för satelliterna efter stöten. Vi tar uttrycken för hastigheterna efter stöten och sätter in våra parametrar [6]. Nu för att få de nya positionerna för satelliterna använder vi likformigt föränderlig rörelse med de nya hastigheterna [7].

```
def update_sat(x1,x2,v1,v2,F,dt):

    global docked

    acc = F/m1 #-------------------------------------------------------------------[1]

    xnew1 = x1 + (v1*dt)

    xnew2 = x2 + (v2 * dt)

    vnew1 = v1 + (acc*dt) # -------------------------------------------------------[2]

    vnew2 = v2

    if abs(xnew2-xnew1)<5:#--------------------------------------------------------[3]

        if abs(vnew2-vnew1)<2: #-----------------------------------------------[4]
            docked = 1
            vnew1 = (m1 * v2 + m2 * v2) / (m1+m2) #--------------------------------[5]
            print("Satellite 1 hastighet innan kollision: ",v1)
            print("Satellite 1 och Satellite 2 hastighet efter kollision: ",vnew1)


        if abs(vnew2-vnew1)>=2: #----------------------------------------------[4]
            docked = 0
            vnew1 = ((m1-m2) * v1 + 2 * m2 * v2) / (m1 + m2) #---------------------[6]
            xnew1 = x1 + (vnew1*dt) #----------------------------------------------[7]
            print("Satellite 1 hastighet innan kollision: ",v1)
            print("Satellite 1 hastighet efter kollision: ",vnew1)
            vnew2 = ((m2-m1) * v2 + 2 * m1 * v1) / (m1 + m2) #---------------------[6]
            xnew2 = x2 + (vnew2*dt) #----------------------------------------------[7]
            print("Satellite 2 hastighet innan kollision: ",v2)
            print("Satellite 2 hastighet efter kollision: ",vnew2)

    return xnew1,xnew2,vnew1,vnew2
```

4 Resultat

Försök	Dockning (Ja/Nej)	Satellit 1 Innan (m/s)	Satellit 1 Efter (m/s)	Satellit 2 Innan (m/s)	Satellit 2 Efter (m/s)
1	Nej	22,8	-7,6	0	15,2
2	Nej	15,9	-5,3	0	10,6
3	Nej	8,1	-2,7	0	5,4
4	Ja	1,87	0,6	0	0,6
5	Ja	0,96	0,39	0	0,39

Vi har infört 5 testfall, tre stycken för när satelliterna inte dockar samt två där de dockar. Vidare kollar vi hastigheten före kollisionen och efter kollisionen på båda satelliterna som vi

infört i tabellen ovan.

5 Diskussion

Det vi kan se utifrån tabellen som är relevant för oss är testfallen där det sker dockning. Vi kan läsa av att satellit 1 måste ha en hastighet på under 2 m/s för att det ska ske dockning. Om relativ hastighet är mindre än 2 m/s kommer satelliterna att anses ha dockat, och deras hastigheter kommer att kombineras enligt principen om fullständigt inelastisk stöt. Båda satelliterna rör sig vidare med samma hastighet. Vilket betyder att det inte är ett isolerat system eftersom hastigheten på satelliter innan kollision är inte samma som efter kollision. Om relativ hastighet är större eller lika med 2 m/s anses dockning misslyckad, och en elastisk stöt simuleras. I detta fall kommer båda satelliternas hastigheter att ändras enligt principen om elastisk stöt, och de kommer att röra sig ifrån varandra.

Referenser

1. "Impuls - Vad är impuls?". FysikStugan. (Åtkomstdatum 17 jan 2024).
2. "Newtons tredje lag". Fysikguiden. (Åtkomstdatum 17 jan 2024).
3. "Newtons andra lag - kraft är lika med massan gånger acceleration". Fysikguiden. (Åtkomstdatum 17 jan 2024).
4. "Impuls". Fysikguiden. (Åtkomstdatum 17 jan 2024).
5. "Rörelsemängd - massa multiplicerat med hastighet". Fysikguiden. (Åtkomstdatum 17 jan 2024).
6. "Stötar & kollisioner". Fysikguiden. (Åtkomstdatum 17 jan 2024).
7. "Lägesenergi och potentiell energi". Fysikguiden. (Åtkomstdatum 17 jan 2024).
8. "Rörelseenergi och kinetisk energi". Fysikguiden. (Åtkomstdatum 17 jan 2024).
9. "Conservation of Energy and Momentum". LibreTexts Physics. (Åtkomstdatum 17 jan 2024).

10. "Likformig föränderlig rörelse". Formelsamlingen. (Åtkomstdatum 17 jan 2024).
11. "Stötar & Acceleration - Ändring av hastighet över tid". Fysikguiden. (Åtkomstdatum 17 jan 2024).

Python-kod

```
"""
Program for simulation of satellite docking between satellite 1 and
satellite 2.

Parameters:
docked = 0      # Flag for controlling if the satellites
                # have docked (docked = 1) or not (docked = 0)

F = 300         # Initial force affecting satellite 1

m1 = 500        # Mass of satellite 1
x1 = -100       # Initial position of satellite 1
v1 = 10         # Initial velocity of satellite 1

m2 = 1000       # Mass of satellite 2
x2 = 0          # Initial position of satellite 2
v2 = 0          # Initial velocity of satellite 2

dt              # Time step. Updated in each iteration and given as
                # the difference in (absolute) time between current iteration
                # and previous iteration.

Task: Modify the function update_sat so that the positions and velocities
of the satellites are updated corrected and obeying the laws of physics.
(see further instructions inside function update_sat).

Last update
Jorgen Ekman, 13 January 2023

Updated by Ömer Kolsuz & Firoz Akbari, 17 January 2024
"""
#import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Button
import time

docked = 0  # Flag for controlling if the satellites
            # have docked (docked = 1) or not (docked = 0)
t_lim = 40  # The duration in seconds the simulation lasts.
            # Should be around 40 s at hand in, but can be changed during
```

```
            # the development of the function.
def update_sat(x1,x2,v1,v2,F,dt):
    """
    Indata:
    x1 (float): position of satellite 1 at time t
    x2 (float): position of satellite 2 at time t
    v1 (float): velocity of satellite 1 at time t
    v2 (float): velocity of satellite 2 at time t
    F (float):  force affecting satellite 1 at time t
    dt (float): time step at time t

    Returnerar:
    xnew1 (float): position of satellite 1 at time t + dt
    xnew2 (float): position of satellite 2 at time t + dt
    vnew1 (float): velocity of satellite 1 at time t + dt
    vnew2 (float): velocity of satellite 2 at time t + dt

    Task: Modify the function so that the positions and velocities of the
    satellites are uppdated correctly and obeying the laws of physics.
    If the distance between the satellites is less than 5 m, one of two things
    will happen:
    1. If the relative velocity is less than 2 m/s, the satellites will dock,
    which is modeled as a totally inelastic collision (fullstandigt inelastisk stot).
    2. If the relative velocity is larger than or equal to 2 m/s, the
    satellites will bounce of each other (docking failed),
    which is modeled as an elastic collision (elastisk stot).
    """
    global docked

    #acc är accelartionen
    #newtons 2:a lag (acceleration = Kraft / massa)
    acc = F/m1

    # xnew1 är ny position för satellite 1.
    # xnew1 => ny position = föregående position + (hastighet * tid)(Likformig föränderlig
rörelse där acc är konstant)
    xnew1 = x1 + (v1*dt)

    #xnew2 är ny position för satellite 2.
    #xnew2 =>  ny position = föregående position * (hastighet + tid)
    xnew2 = x2 + (v2 * dt)

    #vnew1 är den nya hastigheten vid tiden. v(1) är nuvarande hastigheten.
    #Vid konstant acceleration fås hastigheten som v = v0+a*t, där v0 är starthastigheten
    #vnew1 => v(t+dt) = v(1) + acc*dt används för att beräkna hastighetsändringen vid
    # en viss tidpunkt när accelerationen varierar med tiden.
    vnew1 = v1 + (acc*dt)

    #eftersom v2 är på vila mood
    vnew2 = v2

    #If the distance between the satellites is less than 5 m
    if abs(xnew2-xnew1)<5:

    #Rörelsemängd(p) = m(massa) * v(hastighet)
    #Om hastigheten är mindre än 2m/s kommer satelliten docka (fullstandigt inelastisk stöt)
    #I denna typ av kollision förenas de två satelliterna och rör sig vidare med samma hastighet
vilket är 0 m/s.
        if abs(vnew2-vnew1)<2:
            docked = 1
            #vnew1 = hastigheten efter kollision
            #vnew1 = ((m1*vnew1)/(m1+m2))
```

```
            vnew1 = (m1 * v2 + m2 * v2) / (m1+m2)
            print("Satellite 1 hastighet innan kollision: ",v1)
            print("Satellite 1 och Satellite 2 hastighet efter kollision: ",vnew1)

    #Om hastigheten är större eller lika med 2 m/s, då kommer inte docka (elastisk stöt).
    #I denna typ av kollision bevaras både satelliternas rörelsemängd och rörelseenergi.
        if abs(vnew2-vnew1)>=2:
            docked = 0
            #vnew1 = hastigheten efter kollision
            vnew1 = ((m1-m2) * v1 + 2 * m2 * v2) / (m1 + m2)
            xnew1 = x1 + (vnew1*dt)
            print("Satellite 1 hastighet innan kollision: ",v1)
            print("Satellite 1 hastighet efter kollision: ",vnew1)
            vnew2 = ((m2-m1) * v2 + 2 * m1 * v1) / (m1 + m2)
            xnew2 = x2 + (vnew2*dt)
            print("Satellite 2 hastighet innan kollision: ",v2)
            print("Satellite 2 hastighet efter kollision: ",vnew2)

    return xnew1,xnew2,vnew1,vnew2

# Initialisation of some parameters (don't change)
F = 300

m1 = 500
x1 = -100
v1 = 0

m2 = 1000
x2 = 0
v2 = 0

fig, ax = plt.subplots()
# Adjust figure to make room for buttons
fig.subplots_adjust(bottom=0.25)

# Create button which decrease force with 50 N.
decrax = fig.add_axes([0.2, 0.05, 0.2, 0.08])
decr_button = Button(decrax, 'Decrease Thrust', hovercolor='0.975')

def decr(event):
    global F
    F = F - 50.0

decr_button.on_clicked(decr)

# Create button which increase force with 50 N.
incrax = fig.add_axes([0.65, 0.05, 0.2, 0.08])
incr_button = Button(incrax, 'Increase Thrust', hovercolor='0.975')

def incr(event):
    global F
    F = F + 50.0

incr_button.on_clicked(incr)

tstart = time.time()
telapsed = 0
told = tstart
# Main loop startshere
while telapsed <= t_lim:
    # Deduce time and time step
    tnew = time.time()
```

```
dt = tnew - told
told = tnew

# Call to function update_sat
x1,x2,v1,v2 = update_sat(x1,x2,v1,v2,F,dt)

telapsed = time.time() - tstart

# Update plot
ax.plot(x1,0,'wo')
ax.plot(x2,0,'ro',markersize=10)
ax.set_xlabel('x (m)',fontsize=12)
ax.set_xlim([-150,50])
ax.set_facecolor("black")
ax.tick_params(labelsize=12, left = False, labelleft = False)

# Update text
textstr = '\n'.join((
'Time: %6.2f s' % (telapsed,),
'Distance: %6.2f m' % (abs(x2-x1), ),
'Relative  velocity: %6.2f m/s' % (abs(v2-v1), )))
props = dict(boxstyle='round', facecolor='wheat', alpha=1.0)
ax.text(0.25, 0.9, textstr, transform=ax.transAxes, fontsize=12,
    verticalalignment='top', bbox=props)
textstr2 = ('Force: %4.1f N' % (F))
ax.text(0.4, -0.225, textstr2, transform=ax.transAxes, fontsize=12,
    verticalalignment='top')

# If succesful docking
textstring = "Docking succesful!!"
if docked == 1:
    ax.text(0.5, 0.2, textstring, transform=ax.transAxes, color="white", fontsize=12,
    verticalalignment='top')

plt.pause(0.1)

# Don't clear the last plot
if telapsed < t_lim:
    ax.cla()
```
