

HVORDAN COMPUTATIONAL ESSAY PROSJEKTET
BIDRO TIL FORSKNING, UNDERVISNINGSMATRIELL
OG MIN UTVIKLING SOM FYSIKER

Karl Henrik Fredly

PhD Stipendiat ved CCSE, UiO

INNHOOLD

- Hva er et computational essay?
- Computational essay prosjektet

- Min rolle i prosjektet
- Hva jeg produserte
- Hva jeg fikk ut av det

What is the fastest possible volleyball serve?

A computational essay by Karl Henrik Fredly, undergraduate at the University of Oslo (karlhf@student.uv.uio.no)

When I was in high school I really liked playing volleyball. Getting a good serve, spike or block always felt great. Me and my friend Filip used to practice serving after school, hitting the ball back and forth. But his serves were always way better than mine, having much more power and spin. On the court his serves were an absolute pain to return, while mine were just decent. I could never quite figure it out. But maybe I can now? By using my knowledge about physics and computation, can I find out what it takes to make the fastest possible serve? It's worth a shot.

To do the calculations I will use numpy. And to show my results I will use matplotlib.

```
[1]: import numpy as np
import matplotlib.pyplot as plt
```

The rules

In volleyball, the goal is to make the ball hit the floor on your opponents side of the net. The strategy then, is to make it as hard as possible for the opponent to catch the ball, often by setting up for a spike by the net after your team catches the ball. To get the ball into play, you serve the ball from behind the court, over to your opponents side, where they try to catch it. It is beneficial to also make their initial catch of the ball as difficult as possible, either by making the serve fast or unpredictable.

When I say "the fastest possible serve", what I mean is the serve that spends the least amount of time in the air before hitting the opponents floor, giving my opponents the smallest amount of time possible to react to my serve.

Before I tackle the problem, I'll need to define some parameters, as the court, net, ball and air will affect the outcome of the serve. Information about the court, net and ball were taken from the [FIVB](#).

A volleyball court is 18m by 9m, and the net has a height of 2.43m, standing tall in the middle of the court. The ball weighs 270g, and has a radius of 10,5cm.

```
[2]: outX = 18
netX = 9
netheight = 2.43
ballmass = 0.265
radius = 0.105
A = np.pi * radius**2 #Cross sectional area
```



Test run

Before I try to find the optimal velocity and starting angle, I will compute an average serve as a starting off point for further analysis.

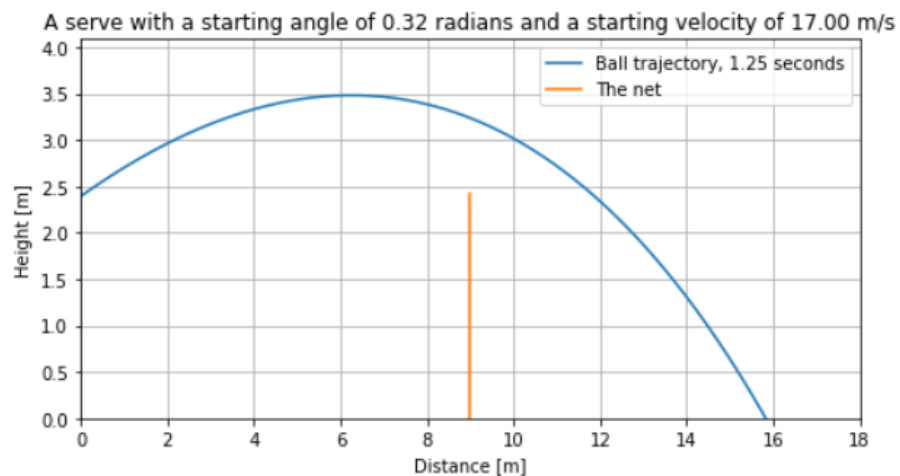
First I define the number of iterations and the time-step for the Euler-Cromer calculation. Then I define the starting velocity and angle, and finally I call my function to compute the serve.

```
[8]: num = 2000
      timestep = 0.001
      v0 = 17 #m/s
      theta0 = 0.32 #starting angle in radians
      examplePosition, exampleTime = serve(v0, theta0, 2000, 0.001)
```

Plotting the example serve:

```
[9]: plt.figure(figsize=(8, 4))
      plt.plot(examplePosition[:, 0], examplePosition[:, 1] - radius, label="Ball trajectory, {:.2f} seconds".format(exampleTime or 0))
      plt.xlim(0, outX)
      plt.ylim(0, np.nanmax(examplePosition[:,1])+0.5)
      plt.plot([9,9],[0,netheight], label="The net")
      plt.grid()
      plt.legend(loc="upper right")

      plt.title("A serve with a starting angle of {:.2f} radians and a starting velocity of {:.2f} m/s".format(theta0, v0))
      plt.xlabel("Distance [m]")
      plt.ylabel("Height [m]")
      plt.show()
```



Now that I have the tools to simulate a serve and see if it was valid, it's time to see how the starting speed and angle affect the trajectory an total airtime.

COMPUTATIONAL ESSAY PROSJEKTET

- Utdanningsforskning
- Forsøkskanin
- Produsering av eksempler

- Elektromagnetisme
- Svingninger og bølger
- 5 publiserte artikler



MIN ROLLE I PROSJEKTET

- Ansatt 2. året i Bachelor
- Produsere eksempler
- Diskutere form og innhold
- Prøve ut flere måter å skrive comp essay



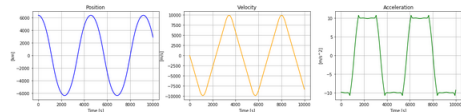
But what if gravity pointed the other way? What if gravity pushed things away from each other? This is a very exciting question with many different avenues to explore (most of them ending with everyone dying), but it will not be the focus of this essay. Gravity will instead only push only you away from Earth.

What this means in practice is that a force given by the formula $F = \frac{GMm}{r^2}$, where M is the mass of the Earth and m is your mass, will push you off the face of planet (given that you're outside).

Just like with ordinary gravity, you mass won't have an impact on your acceleration. Newton's second law together with our formula for anti-gravity gives us that $F = ma = \frac{GMm}{r^2} \Rightarrow a = \frac{GM}{r^2}$

```
G = 6.674 * 10**(-11) #Gravitational constant
M = 5.972 * 10**24 #Mass of the Earth
R_Earth = 6.371 * 10**6 #Radius of the Earth

#This function takes your current height above the ground and returns your c
def antiGravAccel(h):
    return G * M / (R_Earth + h)**2
```

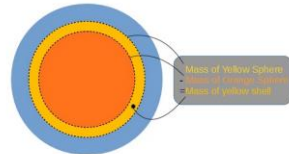


Now I have all that we need to find out how long it takes to fall through the Earth. To find out the time it takes, I need to find out how long it takes to reach the first dip of the position graph, as that is the "bottom" of the Earth.

```
#In order to not find out the time of later dips, I only look at the positio
minIndex = np.argmin(positions[:3000]) #Index of the lowest value of positio
minutes = times[minIndex]/60
exSeconds = times[minIndex]%60
print("The time it takes to travel through the Earth is {:.0f} minutes and {
#time uses seconds, so I divide by 60 to get minutes, and use %60 to get the
```

The time it takes to travel through the Earth is 38 minutes and 10 seconds.

It takes slightly over 38 minutes to fall through the Earth!



Note that I use the same density for the inner and outer distance to calculate the mass of the shell. The density I use for the spheres is the average of the Earth's density at the outer and inner radius of the shell. So that:

$$M_{shell} = M_{outer} - M_{inner} = \frac{4}{3}\pi r_{outer}^3 Density - \frac{4}{3}\pi r_{inner}^3 Density$$

By doing this calculation for every shell and adding them up, I find the total mass of the Earth up to any distance from the center. This process is known as numerical integration using the midpoint rule, and in code it looks like this:

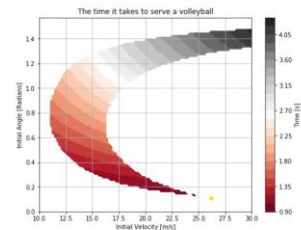
```
pi = np.pi
masses = []
shellMass = 4/3*np.pi*radius[0]**3*density[0] #the innermost shell. In our c
masses.append(shellMass)

for i in range(1, len(radius)):
    shellDensity = (density[i-1]+density[i])/2 #the average of the density at
    #the find the mass of the shells corresponding to every data point
    shellMass = 4/3*np.pi*(radius[i]**3-shellDensity - 4/3*np.pi*radius[i-1]**3)sh
    #the add the shell mass to the list, adding the mass of the current
    masses.append(shellMass + masses[i-1])
```

```
X, Y = np.meshgrid(velocities, angles)
plt.figure(figsize=(8, 8))
ax = plt.contour(X, Y, time, 25, cmap='magma')
cbar = plt.colorbar()
cbar.ax.set_xlabel('Time [s]')

result = np.where(time == np.namin(time))
bvel = velocities[result[0][0]] #best velocity
btheta = angles[result[0][0]] #best angle
plt.plot(bvel, btheta, 'o', color='yellow') #the yellow point on the plot

plt.grid()
plt.title("The time it takes to serve a volleyball")
plt.xlabel("Initial Velocity [m/s]")
plt.ylabel("Initial Angle [Radians]")
plt.show()
```



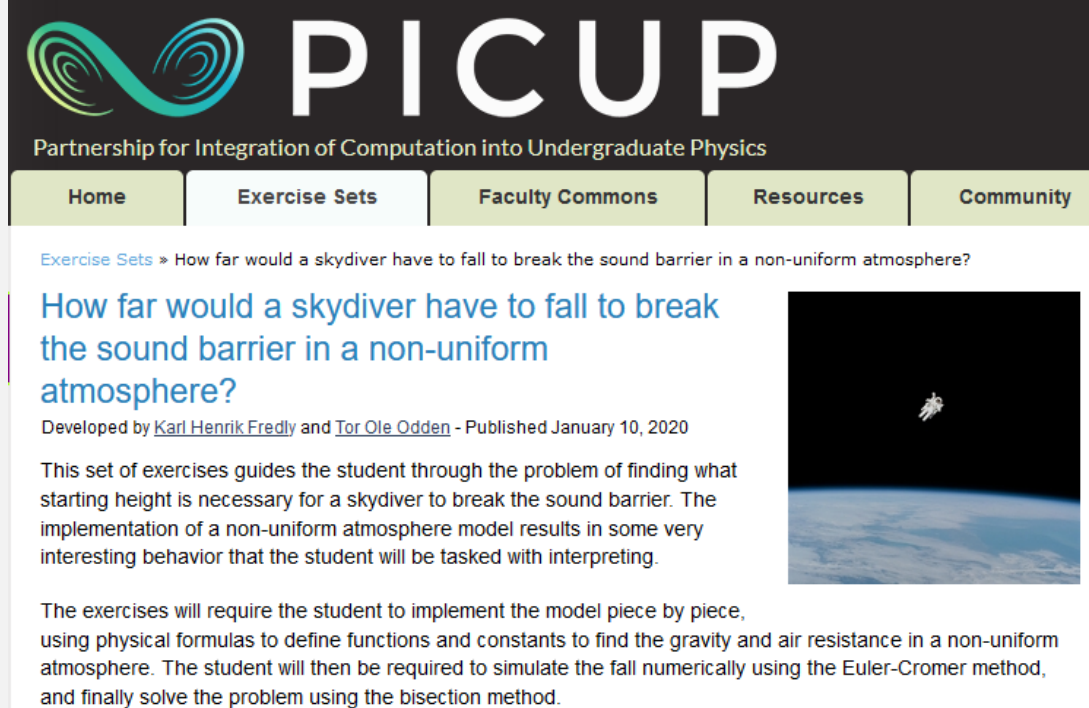
Every possible serve exists on a band which curves from high angle and high velocity (grey area), around low velocity medium angle (pink area), and ends with a sharp point at low angle, high velocity (red area, yellow dot). The yellow dot represents the optimal serve found. Around the optimal serve, there are so few possible serves that they don't show up on the plot at all. As one would expect, there is little room for

HVA JEG PRODUSERTE

- How long would it take to fall through the Earth?
- How long would it take to fall into space?
- What is the fastest possible volleyball serve?

OPPGAVER!

- Deltok i workshop om å produsere oppgaver
- PICUP – Workshops og oppgavedatabase
- To oppgavesett fagfellevurdert og publisert



PICUP
Partnership for Integration of Computation into Undergraduate Physics

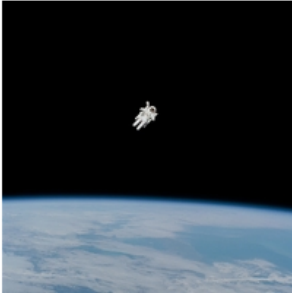
Home Exercise Sets Faculty Commons Resources Community

Exercise Sets » How far would a skydiver have to fall to break the sound barrier in a non-uniform atmosphere?

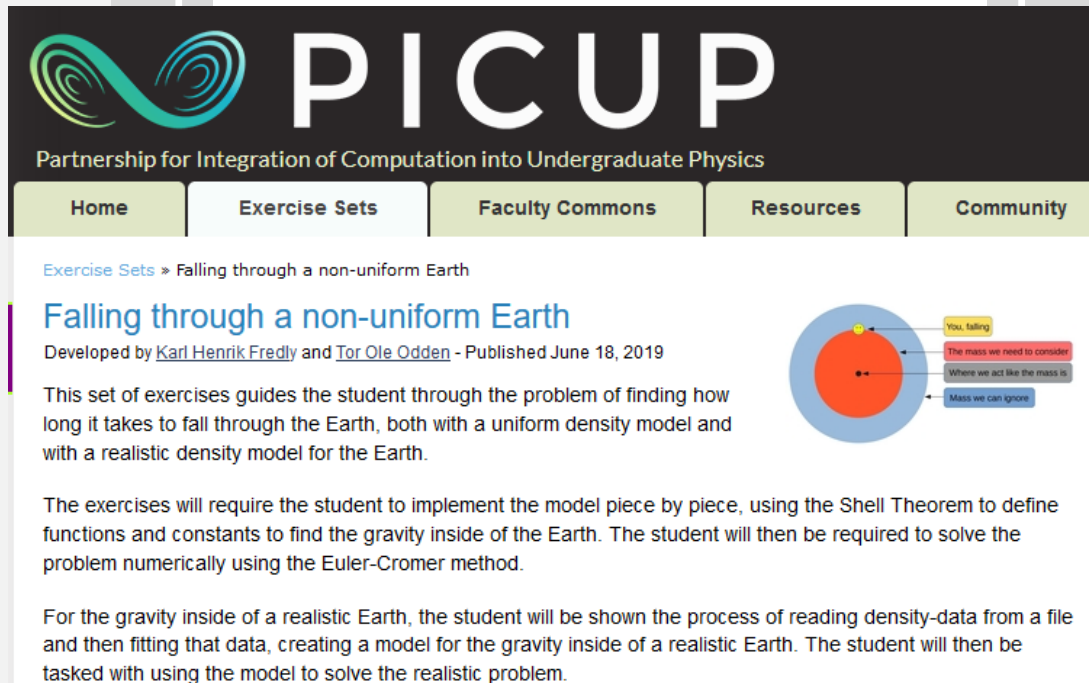
How far would a skydiver have to fall to break the sound barrier in a non-uniform atmosphere?

Developed by [Karl Henrik Fredly](#) and [Tor Ole Odden](#) - Published January 10, 2020

This set of exercises guides the student through the problem of finding what starting height is necessary for a skydiver to break the sound barrier. The implementation of a non-uniform atmosphere model results in some very interesting behavior that the student will be tasked with interpreting.



The exercises will require the student to implement the model piece by piece, using physical formulas to define functions and constants to find the gravity and air resistance in a non-uniform atmosphere. The student will then be required to simulate the fall numerically using the Euler-Cromer method, and finally solve the problem using the bisection method.



PICUP
Partnership for Integration of Computation into Undergraduate Physics


Home Exercise Sets Faculty Commons Resources Community

Exercise Sets » Falling through a non-uniform Earth

Falling through a non-uniform Earth

Developed by [Karl Henrik Fredly](#) and [Tor Ole Odden](#) - Published June 18, 2019

This set of exercises guides the student through the problem of finding how long it takes to fall through the Earth, both with a uniform density model and with a realistic density model for the Earth.



The exercises will require the student to implement the model piece by piece, using the Shell Theorem to define functions and constants to find the gravity inside of the Earth. The student will then be required to solve the problem numerically using the Euler-Cromer method.

For the gravity inside of a realistic Earth, the student will be shown the process of reading density-data from a file and then fitting that data, creating a model for the gravity inside of a realistic Earth. The student will then be tasked with using the model to solve the realistic problem.

HVA JEG FIKK UT AV DET

- Lærte nye ting om:
 - Skrivning
 - Fysikk
 - Å lage oppgaver
- Fot inn døra i CCSE
 - Profag, Honours-emner
- Ble kjent med Tor Ole!
 - Nå veileder for min PhD