DeChambeauDrive

November 13, 2020

[36]:
```
# Conversion Factors
import math

# number
g_mol = 6.02e24
g_mole = g_mol

# distance
g_m=1
g_km=1000*g_m
g_cm=g_m/100
g_mm=g_m/1000.
g_mum=g_m/1e6
g_nm=g_m/1e9
g_pm=g_m/1e12
g_fm=g_m/1e15
g_in=2.54*g_cm
g_ft=12*g_in
g_yard=3*g_ft
g_yards=g_yard
g_mile=5280*g_ft
g_miles=g_mile
g_ly=9460730472580800*g_m
g_Angstrom=1e-10*g_m



# time
g_s=1
g_ms=g_s/1000
g_mus=g_s/1e6
g_min=60*g_s
g_hour=60*g_min
g_hours=g_hour
g_day=24*g_hours
g_days=g_day
g_year=365.25*g_days
```

```
g_years=g_year
g_year_approx=365*g_days
g_month=g_year/12
g_months=g_month

# charge

g_e = 1.602176634e-19 # coulombs

# frequency

g_Hz=1/g_s
g_kHz=1.0e3*g_Hz
g_MHz=1.0e6*g_Hz
g_GHz=1.0e9*g_Hz

# mass
g_kg=1
g_g=g_kg/1000
g_lb=g_kg/2.2
g_oz=(1/16)*g_lb
g_amu = 1.66053906660e-27*g_kg

# Force
g_N = g_kg*g_m/g_s**2
g_dyn = 1e-5*g_N

# Energy
g_J = g_N*g_m
g_kJ = 1.0e3*g_N*g_m
g_W = g_J/g_s
g_kW = 1e3*g_W
g_MW = 1e6*g_W


g_eV=g_e*g_J
g_keV=1e3*g_eV
g_MeV=1e6*g_eV
g_GeV=1e9*g_eV
g_TeV=1e12*g_eV


# Temperature

g_K = 1
g_C = g_K
```

```
# Angles
g_rad=1
g_deg=math.pi/180*g_rad


# Electricity and Magnetism
g_A = g_C/g_s
g_V = g_N*g_m/g_C
g_ohm = g_V/g_A

# Constants of nature
g_c = 2.99792458e8 * g_m/g_s
g_h = 6.62607004e-34 * g_kg*g_m/g_s
g_hbar = g_h/(2*math.pi)
g_G = 6.67408e-11 * g_N*g_m**2/g_kg**2

g_k = 8.88e9*g_N*g_m**2/g_C**2
g_eps0 = 8.85e-12*g_C**2/(g_N*g_m**2)

g_k_b = 1.380649e-23*g_J/g_K # Boltzmann's Constant
g_sb  = 5.670374419e-8*g_W/(g_m**2 * g_K**4)# Stefan-Boltzmann Constant

g_m_proton = 1.67262192369e-27*g_kg
g_m_neutron = 1.674927471e-27*g_kg
g_m_e=9.10910938356e-31*g_kg

g_Ry = 13.605693122994*g_eV
```

[37]:
```
# Rounding to specific precision, from https://github.com/randlet/to-precision
def sigfigs(x,p):
    """
    returns a string representation of x formatted with a precision of p

    Based on the webkit javascript implementation taken from here:
    https://code.google.com/p/webkit-mirror/source/browse/JavaScriptCore/kjs/
 ↪number_object.cpp
    """



    import math
    x = float(x)

    if x == 0.:
        return "0." + "0"*(p-1)

    out = []
    if x < 0:
        out.append("-")
        x = -x

    e = int(math.log10(x))
    tens = math.pow(10, e - p + 1)
    n = math.floor(x/tens)

    if n < math.pow(10, p - 1):
        e = e -1
        tens = math.pow(10, e - p+1)
        n = math.floor(x / tens)

    if abs((n + 1.) * tens - x) <= abs(n * tens -x):
        n = n + 1

    if n >= math.pow(10,p):
        n = n / 10.
        e = e + 1


    m = "%.*g" % (p, n)

    if e < -2 or e >= p:
        out.append(m[0])
        if p > 1:
            out.append(".")
            out.extend(m[1:p])
        out.append('e')
        if e > 0:
            out.append("+")
        out.append(str(e))
    elif e == (p -1):
        out.append(m)
    elif e >= 0:
        out.append(m[:e+1])
        if e+1 < len(m):
            out.append(".")
            out.extend(m[e+1:])
    else:
        out.append("0.")
        out.extend(["0"]*-(e+1))
        out.append(m)

    return "".join(out)
```

```python
def snprint(number, decimals=2):
    template="%%.%sf \\times 10^{%%d}" % (decimals)
    #template='%.{decimals}f \\times 10^{%d}'
    if type(number) is str:
        number_string = number
    else:
        number_string = "%.50e" % (number)
        pass
    [prefix, power10] = number_string.split("e")

    latex_code = "$" + template % (float(prefix), int(power10)) + "$"
    return latex_code
```

1 DeChambeau Drive

During the U.S. Open, professional golfer Bryson DeChambeau hit the ball so hard that his average distance for a drive was 326 yards, a record average distance for a U.S. Open champion. Consider a single drive that hurls the ball 360 yards from the tee. Estimate the maximum speed (in either mph or kph) with which the golf ball must be leaving the tee to accomplish this feat.

[38]:
```python
# Inputs to the calculation

x_1 = 360*g_yard
g = 9.81*g_m/g_s**2
```

1.1 SOLUTION(S)

1.1.1 Background Information

We need some key information before attempting to solve this "Fermi problem." First, we know how far the ball needs to travel. Does it land at the same height as it began? We don't know. We need to make our first assumption: that the ball indeed lands at the same height from which is was launched.

- **ASSUMPTION 1: the ball lands at the same height from which it was launched.**

For the purpose of solving this problem, let us define the vertical coordinate as the y-axis of a Cartesian coordinate system. Let us put the ball at height y=0 at the point of launch and landing. Let us define the line connecting the point of the ball's launch to the place where it lands (before bouncing - we're only concerned with the first time the ball again makes contact with the ground) as the x-axis. Let the starting point of the ball be $x = 0$, and the place where it again makes contact with the ground as $x = 360y$. Let us also convert yards to meters, so $x = 329.18$m. (I am purposefully not rounding to significant figures at this point - save that for the end, to avoid compouding rounding errors)

For now, I will ignore air resistance (I will revisit that later). So here comes assumption 2a:

- **ASSUMPTION 2A: air resistance is negligible in this problem.**

Under that assumption, the only source of acceleration after the ball leaves contact with the driver head will be gravity. Let us put the acceleration due to gravity near the Earth's surface entirely in the negative vertical direction, $a_{gravity}(-\hat{j}) = -g\hat{j}$, with $g = 9.81\text{m/s}^2$. We can then employ the 2-D equations of motion for the ball:

$$
\begin{aligned}
x &= x_0 + v_{0x}t \qquad (1)\\
y &= y_0 + v_{0y}t - \frac{1}{2}gt^2 \qquad (2)
\end{aligned}
$$

The ball will be launched from the tee at an initial angle, with respect to the ground, denoted θ. We can relative the components of the ball's initial velocity, $\vec{v}_0$, to the magnitude and the angle of launch: $v_{0x} = v_0 \cos\theta$ and $v_{0y} = v_0 \sin\theta$.

But at what angle might the ball be launched? This is where we get into the mechanics of drivers.

According to one source, professional drivers are manufacturered to provide their wielders with a specific (a) launch angle and (b) spin rate for the ball (c.f. https://golfweek.usatoday.com/2018/07/02/desired-launch-angle-spin-rates-change-based-on-driver-speed/). The same source suggests that, depending on the speed with which the golfer strikes the ball, a driver should deliver slightly different launch angles because of the energy from the swing that is also transferred into spin. For example, for an 80mph swing the launch angle should be somewhere between 13-14 degrees and the ball back-spin rate should be about 3000rpm. For a 90mph swing speed, the angle should be between 12-14 degrees and the back-spin between 2700-3000 rpm. For a 100mph swing speed, the angle should be between 10-13 degrees and the back-spin between 2300-2700 rpm.

Backspin matters in controlling what the ball does when it lands (c.f. http://large.stanford.edu/courses/2015/ph240/fuster2/), but we can probably ignore it and any ball spin/air iteractions that might, for example, alter the linear kinetic energy of the ball as it travels.

Since we know the range of the ball, excluding some interaction between the back-spin and the drag force on the ball, we don't need (per se) to care about the back-spin for this solution. The range of launch angles, however, will prove valuable.

[39]:
```
# Solution 1 - no air resistance

theta_min = 10*g_deg
theta_max = 14*g_deg

v0_min = math.sqrt(g*x_1/math.sin(2*theta_max))
v0_max = math.sqrt(g*x_1/math.sin(2*theta_min))

v0_avg = (v0_min + v0_max)/2.0
```

```
delta_v0 = math.fabs(v0_avg - v0_min)
delta_v0_pct = 100*delta_v0/v0_avg
```

1.1.2 SOLUTION 1 - IGNORING AIR RESISTANCE

Since the starting and ending heights of the ball are to be the same, we can use the simplified range equation (which also neglects air resistance):

$$x - x_0 = \frac{v_0^2 \sin(2\theta)}{g} \tag{3}$$

For the smallest angle in the given range, one will find the largest initial launch speed; for the largest angle, one will find the smallest initial launch speed.

$$v_0 = \sqrt{\frac{g(x - x_0)}{\sin(2\theta)}} \tag{4}$$

Thus for $\theta_{min} = 10^\circ$ we obtain $v_0^{max} = 97.17\text{m/s}$. For $\theta_{max} = 14^\circ$, we obtain $v_0^{min} = 82.94\text{m/s}$. The average of these is $\overline{v}_0 = 90.05\text{m/s}$. In more familiar units, this is $\overline{v}_0 = 201.4\text{mph}$. If I use the magnitude of the difference between the average speed and the extrema as a measure of "uncertainty" in the number, that yields $\delta v_0 = 7.116$m/s = 15.92mph, or about a 7.90% uncertainty.

1.1.3 CHECKPOINT: DATA FROM BRYSON DECHAMBEAU

A recent article about Bryson DeChambeau noted that after winning the U.S. Open he hit a drive that went over 400 yards with a launch speed of over 200mph (https://sports.yahoo.com/bryson-dechambeau-hits-400-yard-223200497.html):

"Among the eye-popping metrics: 211 mph ball speed and 403.1 yards of carry."

[40]:
```
v0_bd = 211*g_mile/g_hour
d_bd = 403.1*g_yard

theory_ratio = x_1/v0_avg**2
obs_ratio = d_bd/v0_bd**2

# Error propagation on the theory ratio

theory_ratio_error = math.sqrt(4*delta_v0**2/v0_avg**2)*theory_ratio
```

So solution 1, while generally unrealistic, actually gets on the green! Assuming that the basic 2-D motion equation holds (the range equation above) for both our prediction and this observational situation, we can use the ratio $\Delta x/v_0^2$ to compare the two sets of numbers. This yields 0.04059 for our calculation and 0.04143 for the observed numbers. Factoring in the percent uncertainty I estimated on the range of the ball speed, I can do error propagation on this ratio to find that:

$$r \equiv \frac{x}{v_0^2} \longrightarrow \frac{dr}{r} = \sqrt{\frac{dx^2}{x^2} + \frac{4\,dv^2}{v^2}} \tag{5}$$

I choose to ignore uncertainty on the distance, x (that will conservatively underestimate the uncertainty on this ratio). Thus our ratio including this rough uncertainty is $4.1 \times 10^{-2} \pm 6.4 \times 10^{-3}$. That puts the observed ratio and our estimated ratio of these two quantities - speed and range - into essentially plausible agreement.

But what if we factor in air resistance?

[41]:
```
# Inputs to solution 2

rho_air = 1.225 * g_kg/g_m**3
r_ball = 4.268*g_cm
A_ball = math.pi * r_ball**2
C_d = 0.275
m_ball = 45.93 * g_g
```

1.2 SOLUTION 2 - INCLUDING AIR RESISTANCE

This means changing assumption 2A to assumption 2B:

- **ASSUMPTION 2B: air resistance matters**

Relative to the direction of motion of the ball at any one time, the force of drag due to the interaction of the ball with air is given by $F_{drag} = -\frac{1}{2}\rho_{air}C_d A v^2$, where C_d is the drag coefficient of the gold ball, A is the cross-sectional area of the ball, and ρ_{air} is the density of the air. For the latter, we'll assume standard temperature and pressure which yields $\rho_{air} = 1.225$ kg/m^3. A resource on golf balls suggests that the smallest allowed diameter for a golf ball is 4.268cm. (https://www.golfstorageguide.com/golf-ball-size/) This yields a cross-sectional area (assuming the ball is a sphere and the area is a circle) of $A = 5.723 \times 10^{-3}$ m^2.

A regulation golf ball can have a mass not greater than 45.93 grams (https://en.wikipedia.org/wiki/Golf_ball)

What about the drag coefficient for the ball? Golf balls have a dimpled surface, designed to reduce drag, so they won't have the same value of C_d as a smooth sphere (0.47). One research paper (https://www.scirp.org/Journal/PaperInformation.aspx?PaperID=85529) finds that for higher Reynolds Number (low turbulance conditions in the air/ball interaction), experiments find $C_d \approx 0.275$. This leads to the next assumption:

- **ASSUMPTION 3: the air flow over the ball will be less turbulent and more laminar**

Now comes the tricky part. Drag leads to an additional acceleration in the equations of motion, both in the x- and y-directions and dependent on speed of the ball at any given moment. Rather than trying to solve this problem analytically, I will use a numerical solution to the problem:

- Launch the ball with an initial speed v(0)
- Transport the ball over some step in time.
- Compute the change in the velocity due to gravity and drag and update the velocity
- Transport the ball again over the next step in time
- Repeat until y=0 again.

The code below steps in time units of 10 microseconds. In each step, the coordinates of the ball are updated and then the velocity components are corrected using the accelerations. In the x-direction, this is $a_x = F_{drag,x}/m_{ball}$, while in the y-direction it's $a_y = -g + F_{drag,y}/m_{ball}$.

[42]:
```
# Numerical calculation of range
import numpy as np


def CalculateBallRange(v0, theta, C_d):

    vx = v0*math.cos(theta)
    vy = v0*math.sin(theta)

    x = 0
    y = 0

    t_step = 1e-5

    for time in np.arange(t_step,1000, t_step):
        x = x + vx*time
        y = y + vy*time

        # update velocity
        a_drag_x = 0.5 * rho_air * C_d * A_ball * vx**2
        a_drag_y = 0.5 * rho_air * C_d * A_ball * vy**2
        vy = vy - g*time - a_drag_y*time
        vx = vx - a_drag_x*time

        #print(vx, x, vy, y, time)

        if y < 0:
            break

    #print(x,y)
    return x
```

[43]:
```
CalculateBallRange(90*g_m/g_s,14*g_deg, C_d)

# Try a bunch of initial speeds and launch angles and see what gets us close to␣
 ↪the target range
import itertools
v0_list = np.arange(85, 120, 1)
```

```
theta_list = np.arange(10*g_deg, 14*g_deg, 0.5*g_deg)
options = [v0_list, theta_list]
combinations = [o for o in itertools.product(*options)]

closest_combo = None
closest_d = 1e6

v0_spread = 0
v0_n = 0
d_range = 2.5*g_m

for combo in combinations:
    d = CalculateBallRange(combo[0],combo[1], C_d)
    if math.fabs(d - x_1) < d_range:
        print(combo[0],"m/s,", combo[1]/g_deg,"degrees")
        v0_spread += combo[0]
        v0_n += 1
    if closest_combo == None:
        closest_combo = combo
    else:
        if math.fabs(closest_d - x_1) > math.fabs(d - x_1):
            closest_combo = combo
            closest_d = d

v0_spread /= v0_n

print(f"Closest distance to {x_1}m was {closest_d:.2f}m, for {closest_combo}")
```

```
92 m/s, 13.500000000000005 degrees
94 m/s, 13.000000000000004 degrees
95 m/s, 12.500000000000004 degrees
97 m/s, 12.000000000000002 degrees
99 m/s, 11.500000000000002 degrees
101 m/s, 11.000000000000002 degrees
103 m/s, 10.5 degrees
106 m/s, 10.0 degrees
Closest distance to 329.184m was 329.15m, for (92, 0.23561944901923457)
```

[44]:
```
v0_avg = closest_combo[0]
theta_avg = closest_combo[1]
delta_v0 = math.fabs(v0_avg - v0_spread)
delta_v0_pct = delta_v0/v0_avg * 100

theory2_ratio = x_1/v0_avg**2

# Error propagation on the theory ratio
```

```
theory2_ratio_error = math.sqrt(4*delta_v0**2/v0_avg**2)*theory2_ratio
```

A ball speed of 92m/s (206mph), launched at 14 degrees, will get us closest to the target range of 329.184m (a distance of 329m in this case). If we consider all numerical solutions that got us within 2.5m of the target, then the speed and its estimated uncertainty are $\overline{v}_0 = 92.00\text{m/s}$ = 205.8mph and $\delta v_0 = 6.375\text{m/s}$ = 14.26mph, or about a 6.93% uncertainty.

Again, assuming the relationship x/v_0^2 approximately holds as a basis for comparison of the prediction and the observation (even now factoring in air resistance, which complicates the 2-D motion), we obtain 0.03889 for our calculation and 0.04143 for the actual observations. Factoring in the uncertainty I estimated on the range of the ball speed, our predicted ratio is $3.9 \times 10^{-2} \pm 5.4 \times 10^{-3}$. That puts the observed ratio and our estimated ratio of these two quantities - speed and range - into essentially plausible agreement.

1.3 CONCLUSIONS

While still not a very realistic approach, even factoring in air resistance (e.g. I neglected ball backspin), we're definitely getting on the green with these approaches. The hyper-simplistic "neglect everything" approach actually does quite well, predicting a driving velocity of about 200mph when the data from Bryson's actual drives suggests something also around that number (211mph from the impressive drive discussed in that article, but which yielded a longer range in that example). Adding in air resistance, we still obtain something around 206 mph - higher than what was needed in the case of no air resistance (that checks out!) for the same angle of launch.

However, there is clearly more going on here than is captured in this calculation, since for a 211mph drive Bryson can get the ball about 400 yards, 40 yards farther than what we wanted. Clearly, the physics exploration of this question can go deeper.

```
[ ]:
```