9. — PHY 494: Homework assignment (20 points total)

Due Thursday, March 26, 2020, 23:59.

Submission is to your **private GitHub repository**.

Read the following instructions carefully. Ask if anything is unclear.

1. `cd` into your assignment repository (change *YourGitHubUsername* to your GitHub username) and run the update script `./scripts/update.sh` (replace *YourGitHubUsername* with your GitHub username):

   ```
   cd  assignments-2020-YourGitHubUsername
   bash ./scripts/update.sh
   ```

   It should create three subdirectories[1] `assignment_09/Submission`, `assignment_09/Grade`, and `assignment_09/Work`.

2. You can try out code in the `assignment_09/Work` directory but you don't have to use it if you don't want to. Your grade with comments will appear in `assignment_09/Grade`.

3. Create your solution in `assignment_09/Submission`. Use Git to `git add` files and `git commit` changes.

   You can create a PDF, a text file or Jupyter notebook inside the `assignment_09/Submission` directory as well as Python code (if required). **Name your files `hw09.pdf` or `hw09.txt` or `hw09.ipynb`**, depending on how you format your work. Files with code (if requested) should be named exactly as required in the assignment.

4. When you are ready to submit your solution, do a final `git status` to check that you haven't forgotten anything, commit any uncommited changes, and `git push` to your GitHub repository. Check on *your* GitHub repository web page[2] that your files were properly submitted.

[1] If the script fails, file an issue in the Issue Tracker for PHY494-assignments-skeleton and just create the directories manually.

[2] `https://github.com/ASU-CompMethodsPhysics-PHY494/assignments-2020-`*YourGitHubUsername*

You can push more updates up until the deadline. Changes after the deadline will not be taken into account for grading.

Homeworks must be legible and intelligible or may otherwise be returned ungraded with 0 points.

9.1. Advanced Baseball physics (20 points)

In class we wrote code to simulate the trajectory for a curve ball (a summary is provided in Appendix A and you should also consult the Jupyter notebook 12_ODE_applications/baseball_solution.ipynb in the *PHY494-resources* repository[3]). *You are granted permission to use any parts of this notebook for this homework without having to attribute it to the author, Oliver Beckstein, in any way.*

In this problem you should make the simulation more realistic. Include the following improvements and show and discuss in how far they change the results that used a simpler model.

9.1.1. Additional physical effects

1. The quadratic drag coefficient C_D depends on the velocity. In particular, it exhibits a "drag crisis" whereby its aerodynamic drag sharply *decreases* at a critical velocity v_c (Frohlich, 1984) as shown in Figure 1. Wang (2015) parametrizes the dimensionless drag coefficient

[3] As usual, `git pull` in your `PHY494-resources` directory. If you want to use the notebook `baseball_solution.ipynb` as the basis for your solution, then copy it to your `assignments-2020-`*`YourGitHubUsername`*`/assignment_09/Submission` directory and name it `hw09.ipynb`.

$C_D(v)$ as

$$C_D(v) = a + \frac{b}{1+\exp(\chi)} - c \times \begin{cases} \exp(-\chi^2), & \chi < 0, \\ \exp(-\chi^2/4), & \chi \geqslant 0, \end{cases} \tag{1}$$

$$\chi(v) = \frac{1}{4\,\mathrm{m/s}}(v - v_c)$$

$$v_c = 34\,\mathrm{m/s}$$

$$a = 0.36, \qquad b = 0.14, \qquad c = 0.27.$$

Baseballs are pitched in a speed range between 50 mph (for knuckleballs) to 90 mph (fastballs), corresponding to 22 m/s to 41 m/s, which is around the critical velocity $v_c = 34$ m/s for a typical baseball. Therefore, more detailed modelling of C_D might be important to better understand the specific behavior of different pitch types (Frohlich, 1984) (as well as optimum batting parameters (Sawicki et al., 2003)).

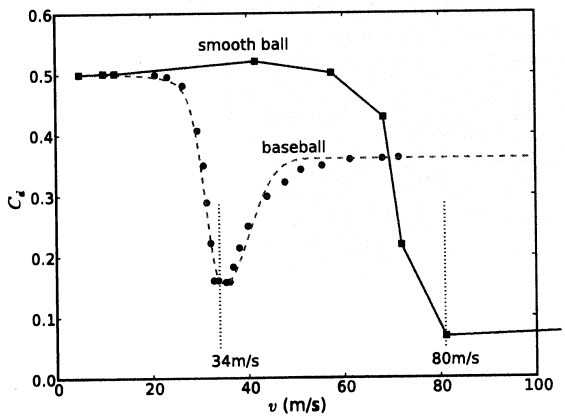

Figure 1. Drag coefficient C_D as a function of velocity for smooth balls (straight line) and baseballs (dashed line). From Wang (2015). Image Copyright ©2015 J. Wang.

2. Due to friction, the ball will not keep spinning at the initial velocity. Instead we can model its slow-down approximately with an exponential decay (Nathan, 2008a) (following Adair (2002))

$$\boldsymbol{\omega}(t) = \boldsymbol{\omega}_0 \exp(-t/\tau). \tag{2}$$

In principle, τ is velocity dependent but here you can make the simplifying assumption $\tau \approx 5\,\mathrm{s}$ (constant). For more details see Nathan (2008a).

9.1.2. Tasks

In your analysis show trajectories of pitches with and without the improved modelling. Comment on the size of the effect.

(a) Write a function `C_D(v)` that implements Eq. 1 and returns the value of the speed-dependent quadractic drag coefficient. Plot $C_D(v)$ over the range $0 \leqslant v \leqslant 80\,\mathrm{m/s}$. **[3 points]**

(b) Write a function `omega_friction(omega0, t)` that implements Eq. 2 to model the slow-down of the spinning ball due to friction. The function should take the initial angular velocity vector $\boldsymbol{\omega}_0$ and the current time t as input and return the current value of $\boldsymbol{\omega}(t)$. Plot $|\boldsymbol{\omega}(t)|$ over the range $0 \leqslant t \leqslant 1\,\mathrm{s}$. **[3 points]**

(c) Update the code in `hw09.py` to include the velocity-dependent drag coefficient $C_D(v)$ and the slowing down of the angular velocity $\boldsymbol{\omega}(t)$ (using your functions from (a) and (b)). Write a function

```
simulate_baseball_advanced(v0, omega0, ...)
```

that returns the trajectory for one pitch with initial velocity $\mathbf{v}_0$ and ball angular velocity $\boldsymbol{\omega}_0$. It should return a $N \times 4$ numpy array where the second axis corresponds to (t, x, y, z). **[7 points]**

(d) Create at least three plots for different choices of the initial velocity (which you should choose yourself) where you compare a simulation

without the added effects to one with the effects of section 9.1.1 included. Plot data for the simple model (see Appendix A) and the advanced model (problem (c)) into the same plots so that one can directly see the effect by visually comparing. Analyze your data by briefly describing your graphs and commenting on the size of the effects. (BONUS: Perform additional analysis as you find necessary.) **[7 points]**

You submission should contain the code for your functions in `hw09.py` and a short write-up with the plots from (a)–(d) and the discussion from (d). The write-up can be a Jupyter notebook.

A. Baseball physics summary

For the simple case (discussed in class, see baseball_solution.ipynb) we want to simulate the trajectory of a *curve ball* in the game of baseball. In this problem we simplify the problem somewhat to only include the essential physical effects:

- Only consider quadratic terms in the air resistance (ignore linear terms) and assume that the drag coefficient b_2 is independent of velocity. (See Problem 9.1 above for a more realistic approach where $b_2(v)$).
- Consider the *Magnus effect* due to spin but assume that the ball spins at constant angular velocity. (See Problem 9.1 above for a more realistic approach where the angular velocity decays with time.)

Quadratic air resistance An approximately quadratic dependence of the drag force on the velocity occurs at high Reynolds numbers, i.e., turbulent

flow (approximately when Re > 2300). An approximate expression is[4]

$$\mathbf{F}_2 = -\frac{1}{2} C_D \rho A v^2 \frac{\mathbf{v}}{v} \tag{3}$$

We are considering the quadratic drag coefficient b_2 to be constant in this problem.

Magnus effect The airflow is changed around a spinning object. The Magnus force is

$$\mathbf{F}_M = \alpha \boldsymbol{\omega} \times \mathbf{v} \tag{4}$$

where $\boldsymbol{\omega}$ is the ball's angular velocity in rad/s (e.g., 200/s for a baseball).

For a sphere the proportionality constant α can be written

$$\mathbf{F}_M = \frac{1}{2} C_L \rho A \frac{v}{\omega} \boldsymbol{\omega} \times \mathbf{v} \tag{5}$$

where C_L is the lift coefficient, ρ the air density, A the ball's cross section. (Advantage of defining C_L this way: when spin and velocity are perpendicular, the Magnus force is simply $F_M = \frac{1}{2} C_L \rho A v^2$.)

C_L is mainly a function of the *spin parameter*[5]

$$S = \frac{r\omega}{v} \tag{6}$$

with the radius r of the ball. $S = v_{\text{spin}}/v$ is the ratio of the speed of a point on the ball's surface to the translational speed of the ball. In general we write

$$\mathbf{F}_M = \frac{1}{2} C_L \frac{\rho A r}{S} \boldsymbol{\omega} \times \mathbf{v} \tag{7}$$

For a baseball, experimental data show approximately a power law dependence of C_L on S

$$C_L = 0.62 \times S^{0.7}. \tag{8}$$

[4] In the Lecture notes we just denoted quadratic drag with $\mathbf{F}_2 = -b_2 v \mathbf{v}$ and lumped everything into the quadratic drag coefficient b_2. Eq. 3 gives more physical motivation to $b_2 = \frac{1}{2} C_D \rho A$

[5] For a more detailed discussion that also considers an additional v-dependence of C_L through its dependence on $C_D(v)$ see Nathan (2008b).

A.1. Baseball equations

In order to simulate the trajectory $\mathbf{r}(t)$ of a baseball, the following equations must be solved:

$$\frac{d\mathbf{r}}{dt} = \mathbf{v} \tag{9}$$

$$\frac{d\mathbf{v}}{dt} = -g\hat{\mathbf{e}}_y - \frac{b_2}{m} v\mathbf{v} + \frac{\alpha}{m}\, \boldsymbol{\omega} \times \mathbf{v} \tag{10}$$

with

$$b_2 = \frac{1}{2} C_D \rho A. \tag{11}$$

The dependence of the dynamical parameters on spin and velocity is

$$\mathbf{F}_M = \alpha\, \boldsymbol{\omega} \times \mathbf{v} \tag{12}$$

$$v = \sqrt{\mathbf{v} \cdot \mathbf{v}} \tag{13}$$

$$S = \frac{r\omega}{v} \tag{14}$$

$$C_L = 0.62 \times S^{0.7} \tag{15}$$

$$\alpha = \frac{1}{2} C_L \frac{\rho A r}{S} \tag{16}$$

A.2. Parameters

Use a ball diameter of 7.468 cm, mass of 148.83 g, and a distance of the pitcher from the home plate $R_{\text{homeplate}} = 18.4\,\text{m}$. Use a constant quadratic drag coefficient $C_D = 0.40$, acceleration due to gravity $g = 9.81\ \text{m} \cdot \text{s}^{-2}$, and density of air $\rho = 1.225\,\text{kg} \cdot \text{m}^{-3}$.

A.3. Simulation

Integrate the equations of motions with the RK4 algorithm (`ode.rk4()`). Stop the integration when $x \geqslant R_{\text{homeplate}}$ or $y < 0.2\,\text{m}$ (i.e., cannot be batted). The integration should be performed inside a function

```
def simulate_baseball(v0, omega, r0,
                      h=0.01, C_D=0.40, g=9.81, rho=1.225,
                      r=0.07468/2, m=0.14883, R_homeplate=18.4):
```

as provided in the skeleton code file `hw09.py`. As input it should take the initial velocity vector $\mathbf{v}_0$ (as a 3D array (v_x, v_y, v_z)), the ball's rotational velocity vector $\boldsymbol{\omega}$ (as a 3D array $(\omega_x, \omega_y, \omega_z)$), and the initial position when leaving the pitcher's hand $\mathbf{r}_0 = (x_0, y_0, z_0)$ (for simplicity, set it to $x_0 = z_{0=}0$ and $y_0 = 2\,\mathrm{m}$).

The function should return the ball's trajectory as an $N \times 4$ array for N time steps along the first axis and $[t, x(t), y(t), z(t)]$ along the second axis (where t is the time and the other three entries are the cartesian components of $\mathbf{r}(t)$).

(a) Simulate a horizontal baseball throw for initial velocity $\mathbf{v} = (30\,\mathrm{m/s}, 0, 0)$. Try out different spins; a good starting value is $\boldsymbol{\omega} = 200\,\mathrm{rad/s} \times 2^{-1/2} \times (0, 1, 1)$. In particular, simulate the baseball throw with

  (i) almost no spin: $\omega = 0.001 \times (0, 0, 1)$ (our code does not handle $\omega = 0$ gracefully...)

  (ii) $\omega = 200 \times (0, 0, 1)$

  (iii) $\omega = 200 \times 2^{-1/2} \times (0, 1, 1)$

(b) Plot the three scenarios in 2D planes: x-y (side view) and x-z (top view). Plot all throws together and add a legend. Briefly describe and discuss the trajectories.

(c) Plot in 3D (see Appendix B).

B. 3D plotting with matplotlib

For the 3D-plotting you can use `matplotlib` as described in the mplot3d Tutorial. In the *jupyter* notebook you can enable interactive view where you can rotate the 3D plot with

```
%matplotlib ipympl
```

(install the *ipympl* package[6] if necessary) or

```
%matplotlib notebook
```

For using `matplotlib` for 3D graphs you need to import `Axes3D` as shown below and then use the `projection='3d'` keyword argument for `add_subplot()`.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(1,1,1, projection='3d')

# add plots for multiple throws: note the order
# of the coordinates
ax.plot(X, Z, Y, 'o', label="no spin")

ax.set_xlabel("$x$ (m)")
ax.set_ylabel("$z$ (m)")
ax.set_zlabel("$y$ (m)")
ax.legend(loc="upper left", numpoints=1)
ax.figure.tight_layout()
```

References

Frohlich C, 1984 Aerodynamic drag crisis and its possible effect on the flight of baseballs. *American Journal of Physics* **52** 325–334, http://dx.doi.org/10.1119/1.13883.

Wang J, 2015 *Computational Modelling and Visualization of Physical Systems with Python* (John Wiley & Sons, Hoboken, NJ).

[6] The *ipympl* package is a matplotlib jupyter extension and improves handling of matplotlib output in jupyter notebooks. It can be installed with `conda install ipympl`. It is preferred over `%matplotlib notebook`.

Sawicki G S, Hubbard M and Stronge W J, 2003 How to hit home runs: Optimum baseball bat swing parameters for maximum range trajectories. *American Journal of Physics* **71** 1152–1162, http://dx.doi.org/10.1119/1.1604384.

Nathan A M, 2008 The effect of spin-down on the flight of a baseball, Tech. rep., University of Illinois, Urbana-Champaign, Urbana, IL, http://baseball.physics.illinois.edu/spindown.pdf.

Adair R K, 2002 *The Physics of Baseball*, 3rd edn. (Harper & Row, Publishers Inc., New York).

Nathan A M, 2008 The effect of spin on the flight of a baseball. *American Journal of Physics* **76** 119–124, http://dx.doi.org/10.1119/1.2805242.