UNIVERSITÀ DEGLI STUDI DI MILANO

Facoltà di Scienze e Tecnologie
Laurea Triennale in Fisica

Simulated run in the rain

Relatore: Prof. Nicola Manini

Cristian Angelo Crespi
Matricola n° 964503
A.A. 2022/2023

Codice PACS: 92.60.jf

Simulated run in the rain

Cristian Angelo Crespi
Dipartimento di Fisica, Università degli Studi di Milano,
Via Celoria 16, 20133 Milano, Italia

February 23, 2024

Abstract

The optimal speed for running in the rain and getting soaked as little as possible has long been discussed in the physics and mathematics community. In practice, however, so far the human body has always been represented as a simple geometric shape such as a parallelepiped or a cylinder. In this work we use numerical simulations address this problem for more complex and dynamically evolving shapes that can represent the human body with an accuracy never reached before in this field.

Advisor: *Prof. Nicola Manini*

Contents

1 Introduction **3**

2 The model **4**

2.1 Analytic results 7

2.1.1 The sphere 7

2.1.2 The parallelepiped 8

2.1.3 The capsule 9

2.2 The human body 11

3 Technical implementation **17**

3.1 Projection surface 17

3.2 Static bodies 19

3.3 Body dynamics 20

3.4 Code validation 22

4 Results **28**

4.1 Error analysis 28

4.2 Optimal velocity 31

5 Discussion and Conclusion **37**

References **38**

1 Introduction

The question of how fast it is convenient to run or walk in the rain is as old as time, and has proven to be a recurring topic in the mathematics and physics community in the last 50 years. Despite the few articles on the topic and the large time span in which they have been produced there have been precious little improvements upon the results first derived by Schwartz and Deakin [1] back in 1973. In that article an orthogonal parallelepiped (from here on we will refer to orthogonal parallelepipeds simply as parallelepipeds as done in previous works on the topic) with sides parallel to the axes is used to approximate a human body running on a straight path in the presence of rain and wind, leading the following results: in the presence of a strong enough tailwind an optimal speed exists, and it equals the component of the rain velocity along the direction of the path; without a tailwind it is always convenient to run faster, although with diminishing returns. Since then a variety of simple geometric shapes have been considered: spheres [2, 3], cylinders [3], ellipsoids [2], plane surfaces [3] and parallelepipeds with generic orientations [3]. The results of these investigations show that the shape considered can change the results considerably: in the presence of a strong enough tailwind a finite optimal speed always exists, but its value depends on the shape considered, and for some shapes an optimal speed can exist even without wind or in the presence of headwind. Because of these discordant results the need for a better approximation of the human body becomes apparent, and the only reason this has not been done yet is the difficulty of deriving analytical results for complex shapes. In the present work we will address this problem by means of a numerical approach. A numerical approach to the problem has been already tried [4], but it too only modeled the human body as a parallelepiped, and its focus was instead on comparing modeling rain as raindrops placed in a cubic lattice as opposed to the more realistic case in which the raindrops are generated at random positions; as expected, the two models agree well.

In this work we consider a model of the body consisting of multiple elementary three-dimensional geometric shapes that move relative to each other, which we will refer to as body parts. We use spheres, parallelepipeds and capsules to model individual body parts. A capsule, also sometimes called a spherocylinder, is an elementary three-dimensional shape consisting in a cylinder with hemispherical ends. Capsules had never been studied before in this context so we had to derive an explicit analytical solution.

2 The model

We model a person running along a straight path under rain and wind. We want to determine is the optimal speed at which she/he should move such that she/he catches the least possible amount of rain. We shall assume the following:

1. The ground is horizontal.
2. The path is rectilinear.
3. The raindrops all have the same size and are densely and uniformly distributed in space.
4. The wind velocity is constant and the raindrops have reached their terminal velocity in the wind.
5. The wind adds a horizontal component to the velocity of the rain.
6. The motion of the body consists of a translation at constant speed along the path plus a periodic relative motion that is generally unique to each body part but shares the same period T.
7. The relative velocity of the body parts due to their periodic motion is negligible compared to the velocity of the rain.
8. The path is long enough that periods of the periodic motions of the body parts are small compared to the time t_f taken to traverse the path ($T \ll t_f$).
9. All involved speeds are negligible compared to the speed of light, so that the nonrelativistic limit and Galilean transformations apply throughout.

In the rest frame of reference, the x axis is aligned with the path of the body and the z axis is be the vertical axis. According to assumptions 1, 2 and 6 the body is translating with a velocity $\mathbf{v}_b = v_b\hat{\mathbf{e}}_x$, and the rain has a velocity $\mathbf{v}_r$. The z component of $\mathbf{v}_r$ is negative, and we refer to its absolute value as the falling velocity v_{fall}. We refer to the value of the y component of $\mathbf{v}_r$ as the "crosswind" v_{cross}. Since all the bodies we take into consideration are symmetric under the transformation $y \to -y$ we consider only positive values of v_{cross}. We refer to the value of the x component of $\mathbf{v}_r$ as the "tailwind" v_{tail}. We call $d = v_b t_f$ the total length of the path traversed.

The water absorbed by the traveler W, measured as the total volume of water that has hit her/him at the end of the walk, is equal to the number of raindrops hit times the volume of each drop. Thanks to our assumption 3 we can safely define a dimensionless "rain density" $\rho_{rain} = N_{drop}V_{drop}/V$, where N_{drop} is the number of raindrops contained

in a certain volume V, and V_{drop} is the volume of a single drop. In short ρ_{rain} is the ratio between the amount of rain contained in a volume and that same volume, i.e. the volume fraction of liquid water freely falling in the atmosphere. Let us consider the frame of reference in which the raindrops are still. In this frame the velocity of the body is:

$$\mathbf{v}_{rel} = \mathbf{v}_b - \mathbf{v}_r. \tag{1}$$

The stationary raindrops inside the volume of space that our body passes through, which we call V_b, are swept up by the body motion and contribute to the water absorbed W. We can then use the following relation to evaluate the wetness: $W = \rho_{rain} V_b$. Since ρ_{rain} is constant in the walk in the rain our problem reduces to evaluating and minimizing V_b. As derived in Ref. [2] for a body whose only movement is a rigid translation V_b is given by the following formula:

$$V_b(\mathbf{v}_{rel}) = S_b(\mathbf{v}_{rel}) \, \frac{\|\mathbf{v}_{rel}\|}{v_b} \, d. \tag{2}$$

$S_b(\mathbf{v}_{rel})$ is the area of the projection of the body on a plane perpendicular to $\mathbf{v}_{rel}$. The only non-trivial part of the problem is then the determination of the dependence of S_b on $\mathbf{v}_{rel}$. Note that generally $\lim_{v_b \to +\infty} V_b(\mathbf{v}_{rel}) \neq 0$; as v_b approaches infinity, $\mathbf{v}_{rel}$ approaches $v_b \hat{\mathbf{e}}_x$, so we get:

$$\lim_{v_b \to +\infty} V_b(\mathbf{v}_{rel}) = S_b(\hat{\mathbf{e}}_x) \, d. \tag{3}$$

This shows that even if there is no finite optimal speed there is a minimum wetness that is not avoidable by going faster, which is independent of the rain velocity v_r. Furthermore since $\lim_{v_b \to 0^+} S_b(\mathbf{v}_{rel}) > 0$ (which is a reasonable assumption for a human body) then $V_b(\mathbf{v}_{rel})$ diverges to infinity as v_b tends to 0 because of the v_b term in the denominator in Eq. (2), so there usually is no upper limit to the wetness at small speed.

We now generalize these findings for a dynamical body. Since the velocity of the body parts is small compared to the velocity of the rain 7 we can approximate the relative velocity of the body parts with the relative velocity of the whole body $\mathbf{v}_{rel}$. This way we can write the area of the projection of the whole body as depending only on the relative velocity of the whole body and of the instantaneous orientation of each body part, which only depend on time for a set body: $\tilde{S_b}(\mathbf{v}_{rel}, t)$. We can then generalize Eq.(2) by considering that $d/v_b = t_f$ and substituting $S_b(\mathbf{v}_{rel}) \, t_f$ with an integral of $\tilde{S_b}(\mathbf{v}_{rel}, t)$ over $[0, t_f]$:

$$V_b = \|\mathbf{v}_{rel}\| \int_0^{t_f} dt \, \tilde{S_b}(\mathbf{v}_{rel}, t). \tag{4}$$

Furthermore since the time spent in the path is long compared to the period of the function $\tilde{S_b}(\mathbf{v}_{rel}, t)$ we can approximate it with its time average over the period T. For

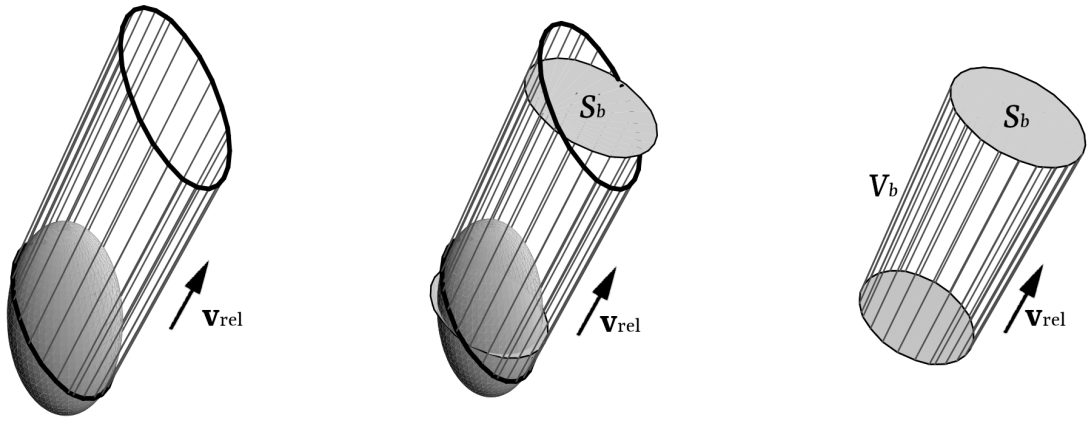

Figure 1: Example with an ellipsoid as the body. Its orthogonal projection on the plane perpendicular to $\mathbf{v}_{rel}$ is an elliptic disk with area S_b and V_b is the volume of the resulting elliptic cylinder. Original image from Ref. [2].

a dynamic body we can then employ the time average

$$S_b(\mathbf{v}_{rel}) = \frac{1}{T}\int_0^T dt\, \tilde{S_b}(\mathbf{v}_{rel}, t) \tag{5}$$

in Eq. (2), which is thus extended to dynamical bodies too, provided that assumptions 6, 7, and 8 are satisfied.

Note that the results are invariant under the transformation $\mathbf{v}_{rel} \longrightarrow -\mathbf{v}_{rel}$, since a plane perpendicular to $\mathbf{v}_{rel}$ is also perpendicular to $-\mathbf{v}_{rel}$. Since we will not be able to solve the dynamic body analytically we will approximate $S_b(\mathbf{v}_{rel})$ with a discrete time average:

$$S_b(\mathbf{v}_{rel}) \simeq \frac{1}{N}\sum_{i=0}^{N-1} \tilde{S_b}\left(\mathbf{v}_{rel},\ i\frac{T}{N}\right). \tag{6}$$

Furthermore, since d is a given, we will evaluate and minimize the ratio of wetting volume to path length instead, or wetness for short, which can be written as:

$$R_b(\mathbf{v}_{rel}) = \frac{V_b(\mathbf{v}_{rel})}{d} = S_b(\mathbf{v}_{rel})\,\frac{\|\mathbf{v}_{rel}\|}{v_b}, \tag{7}$$

which for a given $\mathbf{v}_r$ can be rewritten using Eq. (1) and $\mathbf{v}_b = v_b\hat{\mathbf{e}}_x$ as:

$$R_b(v_b) = R_b(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r), \tag{8}$$

Considering Eq. (3) we can also write:

$$\lim_{v_b\to+\infty} R_b(v_b) = S_b(\hat{\mathbf{e}}_x). \tag{9}$$

The optimal velocity v_{opt} we are looking for is then the value of v_b that minimizes $R_b(\mathbf{v}_{rel})$ for a given $\mathbf{v}_r$. For the sake of convenience we will use v_{fall} as a unit of measure for all velocities in this problem. For R_b we will adopt SI units.

2.1 Analytic results

The elementary geometric shapes we adopt as building blocks for the human body are spheres, parallelepipeds and capsules. To verify if our numerical code works properly it is useful to check numerical results against the analytic solutions. Furthermore, we need to derive the appropriate formulas for the projection of the adopted building blocks onto arbitrary planes. First we define the orthogonal projection onto planes perpendicular to a generic vector $\mathbf{v}$. For simplicity's sake we consider planes passing through the origin. The projector operator $P_{\mathbf{v}}$ onto a plane through the origin and perpendicular to $\mathbf{v}$ acts on vector $\mathbf{a}$ as follows [6]:

$$P_{\mathbf{v}}(\mathbf{a}) = \mathbf{a} - \frac{\mathbf{a} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}}\mathbf{v}. \tag{10}$$

Considering bodies B defined as sets of points in $\mathbb{R}^3$ we define their projection as $P_{\mathbf{v}}(B) := \{P_{\mathbf{v}}(\mathbf{b}) : \mathbf{b} \in B\}$.

2.1.1 The sphere

The sphere is the easiest body to work with since its projection does not depend on $\mathbf{v}$: the orthogonal projection of a sphere of radius r and center $\mathbf{c}$ on any plane is a disk with radius r and center $P_{\mathbf{v}}(\mathbf{c})$ [2, 3]. Obviously, its area is, $S_s = \pi r^2$. Plugging this result into Eq. (7), we obtain the following formula for $R_s(\mathbf{v}_{rel})$:

$$R_s(\mathbf{v}_{rel}) = \pi r^2 \, \frac{\|\mathbf{v}_{rel}\|}{v_b}. \tag{11}$$

Using Eq. (1) and $\mathbf{v}_b = v_b \hat{\mathbf{e}}_x$ we can write $R_s(v_b)$ for any given $\mathbf{v}_r$:

$$R_s(v_b) = \pi r^2 \, \frac{\|(v_b \hat{\mathbf{e}}_x - \mathbf{v}_r)\|}{v_b}. \tag{12}$$

Obviously as $v_b \to +\infty$ we get:

$$\lim_{v_b \to +\infty} R_s(v_b) = \pi r^2. \tag{13}$$

As derived in Refs. [2, 3], for a sphere the condition for v_{opt} to exist finite is simply $v_{tail} > 0$. If there is any tailwind at all there is an optimal velocity, and its value is:

$$v_{opt} = \frac{\|\mathbf{v}_r\|^2}{v_{tail}}. \tag{14}$$

2.1.2 The parallelepiped

As derived in Refs. [1, 3], when dealing with a parallelepiped one only needs to consider the faces that get wet, which are one to three depending on orientations of the parallelepiped and of $\mathbf{v}$. A parallelepiped is defined by its center $\mathbf{c}$ and its three sides $\mathbf{s}_1$, $\mathbf{s}_2$ and $\mathbf{s}_3$, where $\mathbf{s}_i \cdot \mathbf{s}_j = 0$ for $i, j = 1, 2, 3$ and $i \neq j$. The 6 points corresponding to the centers of its 6 faces are $\mathbf{p}_i^{\pm} = \mathbf{c} \pm \mathbf{s}_i/2$. We consider the surface of a face as pointing out of the parallelepiped. Then the face with center $\mathbf{p}_i^{\pm}$ will have surface $\mathbf{S}_i^{\pm} = \pm\left\|\mathbf{s}_j \times \mathbf{s}_k\right\| \hat{\mathbf{s}}_i$, with $\hat{\mathbf{s}}_i = \mathbf{s}_i/\|\mathbf{s}_i\|$ and i, j, k all different. In the system of reference in which the rain is still, the parallelepiped translates with velocity $\mathbf{v}_{rel}$, and it is easy to see that a face $\mathbf{S}_i^{\pm}$ gets wet if and only if the velocity of the body is positive in the direction the face is pointing to, i.e. iff $\mathbf{S}_i^{\pm} \cdot \mathbf{v}_{rel} > 0$, which is equivalent to $\pm\mathbf{s}_i \cdot \mathbf{v}_{rel} > 0$. We can immediately see that if $\mathbf{S}_i^{\pm}$ gets wet then $\mathbf{S}_i^{\mp}$ does not, and iff $\mathbf{s}_i \cdot \mathbf{v}_{rel} = 0$ neither of $S_i^{\pm}$ gets wet, so at most 3 faces can get wet. Since $\mathbf{s}_i$ are 3 orthogonal vectors, they form a complete basis of $\mathbb{R}^3$, and since $\mathbf{v}_{rel}$ is never a null vector, then $\mathbf{s}_i \cdot \mathbf{v}_{rel} \neq 0$ for at least one i.

We now want to evaluate the projections of the wet faces on a plane perpendicular to $\mathbf{v}_{rel}$. The projection of a face is a parallelogram with as vertices the projections of the vertices of the face, and then as sides the projections of the sides of the face. The area of said parallelogram is:

$$A_i(\mathbf{v}_{rel}) = \left\|(\mathbf{s}_j \times \mathbf{s}_k) \cdot \hat{\mathbf{v}}_{rel}\right\|, \tag{15}$$

with $\hat{\mathbf{v}}_{rel} = \mathbf{v}_{rel}/\|\mathbf{v}_{rel}\|$. Notice that if we have three wet faces $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3$ each has two sides in common with too others and they all share the vertex connecting the shared sides. The projection of those three faces will then consist of three parallelograms that each share two sides and a vertex as the original faces. This results in an irregular hexagon with an area equal to the sum of the areas of the 3 composing parallelograms. Its area, which corresponds to $S_p(\mathbf{v}_{rel})$ then is:

$$S_p(\mathbf{v}_{rel}) = \|\mathbf{S}_1 \cdot \hat{\mathbf{v}}_{rel}\| + \|\mathbf{S}_2 \cdot \hat{\mathbf{v}}_{rel}\| + \|\mathbf{S}_3 \cdot \hat{\mathbf{v}}_{rel}\|, \tag{16}$$

with $\mathbf{S}_i = \mathbf{s}_j \times \mathbf{s}_k$. This formula holds even if less than three faces get wet since if $\mathbf{s}_i \cdot \mathbf{v}_{rel} = 0$ then $\left\|(\mathbf{s}_j \times \mathbf{s}_k) \cdot \mathbf{v}_{rel}\right\| = 0$, indicating that the contribution by degenerate faces that do not get wet vanishes. We can then use Eq. (7) find $R_p(\mathbf{v}_{rel})$:

$$R_p(\mathbf{v}_{rel}) = \left[\sum_{i=1}^{3} \|\mathbf{S}_i \cdot \mathbf{v}_{rel}\|\right] \frac{\|\mathbf{v}_{rel}\|}{v_b}. \tag{17}$$

Using Eq. (1) and $\mathbf{v}_b = v_b\hat{\mathbf{e}}_x$ we can write $R_p(v_b)$ for any given $\mathbf{v}_r$:

$$R_p(v_b) = \left[\sum_{i=1}^{3} \|\mathbf{S}_i \cdot (v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)\|\right] \frac{\|v_b\hat{\mathbf{e}}_x - \mathbf{v}_r\|}{v_b}. \tag{18}$$

And as $v_b \to +\infty$ we get:

$$\lim_{v_b \to +\infty} R_p(v_b) = \sum_{i=1}^{3} \|\mathbf{S}_i \cdot \hat{\mathbf{e}}_x\|. \tag{19}$$

Despite the fact that the parallelepiped is the first solid to ever be considered as a model for the human body, analytical solutions have only been derived for parallelepipeds with sides parallel to the axes. In this case we call S_x, S_y and S_z the area of the face perpendicular to the x, y and z axis respectively. Then, as derived in Refs. [1, 3], v_{opt} exists finite under the following condition:

$$v_{tail} > \frac{S_y |v_{cross}| + S_z \, v_{fall}}{S_x}. \tag{20}$$

If this condition is satisfied v_{opt} is always equal to v_{tail}. Immediately we can notice how using a different type of solid to model the human body can change the results: both the existence condition for $v_{opt}$and its value are different for the sphere and the parallelepiped. The only constant is the presence of a tailwind as a necessary condition for the existence of v_{opt}.

2.1.3 The capsule

As stated before, while spheres and parallelepipeds have already been studied in this context, capsules (see Fig. 2) are a new shape, so we provide here a simple analytical solution. A capsule is defined as follows [5]: let L be a line segment in $\mathbb{R}^3$ and r a positive real number. The capsule with axis L and radius r is the set of points whose distance from L is smaller or equal to r. The same capsule can also be defined as the Minkowski sum between L and a ball centered at the origin with radius r. Given two sets of vectors A and B in Euclidean space, their Minkowski sum is the set of points $A + B := \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A, \, \mathbf{b} \in B\}$ [7]. We now proceed to find the orthogonal projection of a capsule C on a plane perpendicular to a generic vector $\mathbf{v}$. We show that $P_{\mathbf{v}}$ distributes over the Minkowski sum, i.e. $P_{\mathbf{v}}(A + B) = P_{\mathbf{v}}(A) + P_{\mathbf{v}}(B)$. Using Eq. (10):

$$P_{\mathbf{v}}(\mathbf{a} + \mathbf{b}) = \mathbf{a} + \mathbf{b} - \frac{(\mathbf{a} + \mathbf{b}) \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \mathbf{v} = \mathbf{a} - \frac{\mathbf{a} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \mathbf{v} + \mathbf{b} - \frac{\mathbf{b} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \mathbf{v} = P_{\mathbf{v}}(\mathbf{a}) + P_{\mathbf{v}}(\mathbf{b}), \tag{21}$$

$$\begin{aligned} P_{\mathbf{v}}(A + B) &= \{P_{\mathbf{v}}(\mathbf{a} + \mathbf{b}) : \mathbf{a} \in A, \, \mathbf{b} \in B\} = \\ &= \{P_{\mathbf{v}}(\mathbf{a}) + P_{\mathbf{v}}(\mathbf{b}) : \mathbf{a} \in A, \, \mathbf{b} \in B\} = \\ &= \{\mathbf{a}' + \mathbf{b}' : \mathbf{a}' \in P_{\mathbf{v}}(A), \, \mathbf{b}' \in P_{\mathbf{v}}(B)\} = P_{\mathbf{v}}(A) + P_{\mathbf{v}}(B). \end{aligned} \tag{22}$$

Incidentally, note that while we are only considering vectors in $\mathbb{R}^3$ and a surface in $\mathbb{R}^2$ equations (21) and (22) hold up for vectors in the Euclidean space $\mathbb{E}^n$ and hypersurfaces

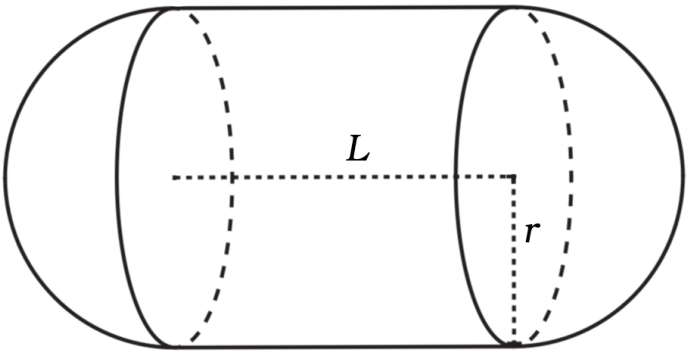

Figure 2: A capsule with axis L and radius r.

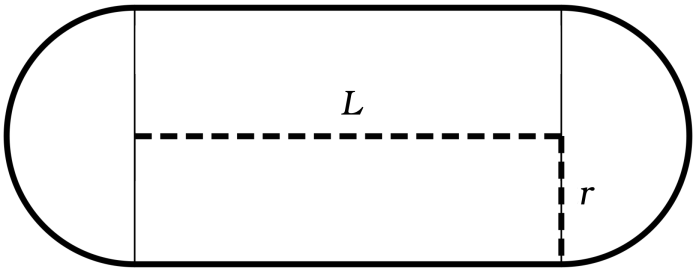

Figure 3: A stadium with axis L and radius r.

in $\mathbb{E}^{n-1}$ for any n. We know that the projection of a line segment L defined by its two endpoints $\mathbf{l}_1$ and $\mathbf{l}_2$ is the line segment $\hat{L} = P_{\mathbf{v}}(L)$ with endpoints $P_{\mathbf{v}}(\mathbf{l}_1)$ and $P_{\mathbf{v}}(\mathbf{l}_2)$, and that the projection of a sphere S with radius r centered at the origin is a disk $\hat{S}$ with radius r also centered at the origin and laying on the projection plane. The projection of our capsule $C = L + S$ then is $\hat{C} = \hat{L} + \hat{S}$. The Minkowski sum between a line segment and a disk is the stadium [8], also called sausage body, a two-dimensional geometric shape defined exactly like a capsule, but in 2D. The stadium with axis L and radius r is the set of points whose distance from L is smaller or equal to r. Revolving a stadium around L results in a capsule. An example of a stadium with radius r and axis L can be seen in Fig. 3. The area of a stadium can be easily derived considering that it is made up of two half disks with radius r plus a rectangle with sides $2r$ and l, where by l we indicate the length of L, i.e. the Euclidean distance between its two endpoints. The area is:

$$A = \pi r^2 + 2rl. \tag{23}$$

The stadium we get from projecting a capsule then has the same radius r, and $\hat{L} = P_{\mathbf{v}}(L)$. Defining $\Delta\mathbf{l} := \mathbf{l}_2 - \mathbf{l}_1$ we can find the length of $\hat{L}$ as:

$$\hat{l} = \|P_{\mathbf{v}}(\mathbf{l}_2) - P_{\mathbf{v}}(\mathbf{l}_1)\| = \|P_{\mathbf{v}}(\Delta\mathbf{l})\| = \left\|\Delta\mathbf{l} - \frac{\Delta\mathbf{l}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v}\right\|. \tag{24}$$

We now have all we need to calculate the projection S_c of our capsule on a plane perpendicular to the relative velocity of the rain $\mathbf{v}_{rel}$. Using equations (23) and (24) we write:

$$S_c(\mathbf{v}_{rel}) = \pi r^2 + 2r\left\|\Delta\mathbf{l} - \frac{\Delta\mathbf{l}\cdot\mathbf{v}_{rel}}{\mathbf{v}_{rel}\cdot\mathbf{v}_{rel}}\mathbf{v}_{rel}\right\|. \tag{25}$$

Substituting in Eq. (7) gives us the ratio R_c:

$$R_c(\mathbf{v}_{rel}) = \left[\pi r^2 + 2r\left\|\Delta\mathbf{l} - \frac{\Delta\mathbf{l}\cdot\mathbf{v}_{rel}}{\mathbf{v}_{rel}\cdot\mathbf{v}_{rel}}\mathbf{v}_{rel}\right\|\right]\frac{\|\mathbf{v}_{rel}\|}{v_b}. \tag{26}$$

Using Eq. (1) and $\mathbf{v}_b = v_b\hat{\mathbf{e}}_x$ we can write $R_c(v_b)$ for any given $\mathbf{v}_r$:

$$R_c(v_b) = \left[\pi r^2 + 2r\left\|\Delta\mathbf{l} - \frac{\Delta\mathbf{l}\cdot(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)}{(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)\cdot(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)}(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)\right\|\right]\frac{\|(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)\|}{v_b}. \tag{27}$$

And as $v_b \to +\infty$ we get:

$$\lim_{v_b\to+\infty} R_c(v_b) = \pi r^2 + 2r\,\|\Delta\mathbf{l} - (\Delta\mathbf{l}\cdot\hat{\mathbf{e}}_x)\,\hat{\mathbf{e}}_x\|. \tag{28}$$

2.2 The human body

Considering the wide variety of body shapes that human beings can exhibit it would seem like an impossible task to find general results for them, and it probably is, so in this work we limit ourselves to building one specific model of a human body. To make sure that our model can properly approximate a real human we have decided to build it using the Vitruvian Man, by Leonardo da Vinci [9] as a guideline. By analysing the text accompanying the drawing and measuring the drawing itself[1] we find the following proportions to follow, given a body with a total height h:

- The head is $h/8$ tall.
- The neck is $h/24$ tall and $h/14$ thick.
- The torso is $h/3$ tall, $h/6$ wide and $h/12$ thick.
- The shoulders are each $h/24$ long and $h \cdot 0.06$ thick.
- The upper arms are each $h/8$ long and $h/20$ thick.

[1]We measured a scan of the original drawing using the software ImageJ [10].

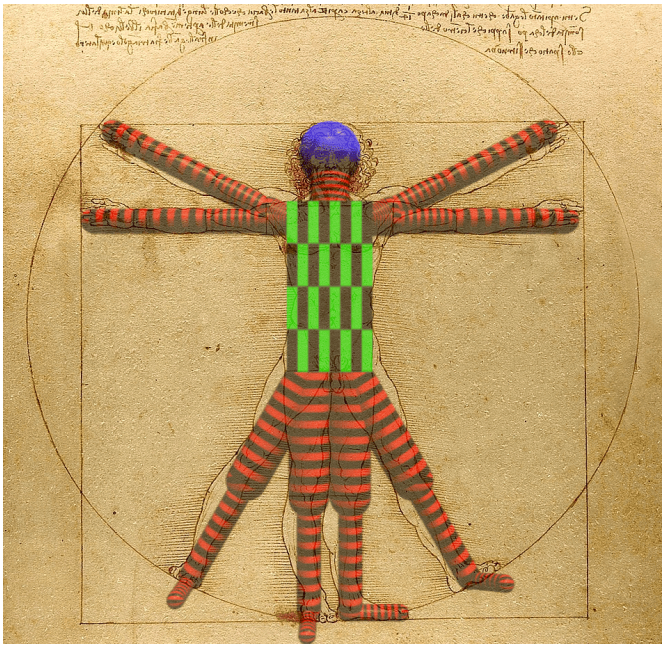

Figure 4: Our model of the human body overlayed on the Vitruvian Man by Leonardo da Vinci. The colors and patterns denote the shape of the base body part: solid blue for sphere, checkered green and black for parallelepiped, and red and black stripes for capsule.

- The forearms (including the hands) are each $h/4$ long and $h/20$ thick.
- The thighs are each $h/4$ long and $h/12$ thick.
- The calves are each $h/4$ long and $h/20$ thick.
- The feet are each $h/7$ long and $h/30$ thick.

We adopt $h = 1.68$ m as a plausible height for a person. While the total wetting scales as h^2, the optimal speed should not depend on the size of the body but only on its shape.

As anticipated we built our body using as base shapes spheres, parallelepipeds and capsules. For the torso we use a parallelepiped with sides $\mathbf{s}_1 = (1/12, 0, 0)\ h$, $\mathbf{s}_2 = (0, 1/6, 0)\ h$, $\mathbf{s}_3 = (0, 0, 1/3)\ h$. For the neck we use a capsule of radius $r = 1/24\ h$ and length $\|\Delta\mathbf{l}\| = 5/48\ h$. The head is modeled with a sphere of radius $r = 1/16\ h$. The neck connects the torso to the head: one endpoint $\mathbf{l}_1$ is placed on the center of the top face of the torso, while the other coincides with the center of the head, i.e. $\mathbf{l}_2 = \mathbf{c}$. Both the neck and the head are still relative to the torso. The shoulders, upper arms, forearms, thighs, calves, and feet are all modeled by capsules with various radii and lengths. The shoulders have length $\|\Delta\mathbf{l}\| = 1/24\ h$ and radius $r = 0.03\ h$. Each of

them has one of its endpoints situated on one of the lateral faces of the torso, 0.03 h below the middle of the top side of the face, and their axis is perpendicular to said face. The second endpoint of each shoulder coincides with the first endpoint of one of the two capsules of length $\|\Delta\mathbf{l}\| = 1/8\ h$ and radius $r = 1/40\ h$ that represent the upper arms. The second endpoint of each upper arm coincides with the first endpoint of one of the two capsules of length $\|\Delta\mathbf{l}\| = 9/40\ h$ and radius $r = 1/40\ h$ that represent the forearms. The shoulders are still relative to the torso during both walking and running. The upper arms can rotate around the endpoint connecting them to the shoulders. This movement is shared by the forearms which can further rotate around the endpoint they share with the upper arms. The thighs have length $\|\Delta\mathbf{l}\| = 1/4\ h$ and radius $r = 1/24\ h$. They both have one of their endpoints on the bottom face of the torso. Their other endpoint coincides with the first endpoint of the two capsules of length $\|\Delta\mathbf{l}\| = 9/40\ h$ and radius $r = 1/40\ h$ that represent the calves. The feet have length $\|\Delta\mathbf{l}\| = 23/210\ h$ and radius $r = 1/60\ h$. They are connected to the calves by one of their endpoint, positioned $r_{calf} - r_{foot} = 1/120\ h$ below the endpoint of each calf, so that the calf does not protrude below the foot. The thighs can rotate along with the calves and feet around the endpoint connecting them to the torso. The calves and the feet can rotate together around the endpoint connecting the calves to the thighs, and the feet can rotate around the endpoint connecting them to the calves. A direct comparison between our model of the body and the Vitruvian Man is shown in Fig. 4. The numerical details of our body are available on GitHub [15].

Given the model details, we now describe how it moves. We study two regimes of motion: walking and running. We model both walking and running with periodic rototranslations of the limbs around their respective joints. For simplicity's sake we approximate the periodic time evolution of the angles θ_j of the joints as sinusoids:

$$\theta_j(t) = \frac{\theta_{max} + \theta_{min}}{2} + \frac{\theta_{max} - \theta_{min}}{2} \sin\left(2\pi \frac{t}{T} + \phi\right). \tag{29}$$

To make sure our dynamic body resembles a real human being we refer to articles on the biomechanics of walking and running for the amplitudes and relative phases of these sinusoids. By convention, we consider for each joint the angle $\theta_j = 0$ when the body is standing with straight legs and the arms parallel to the body, feet perpendicular to the legs. We further assume that $\theta_j > 0$ when a limb moves forward along the direction of motion of the body:

- During walking θ_{hip} ranges from $\theta_{min} = -10°$ to $\theta_{max} = 30°$[12], while during running θ_{hip} ranges from $\theta_{min} = -30°$ to $\theta_{max} = 30°$[13].
- During both walking and running the angle of the leg at the knee θ_{kne} has a relative phase of around $-\pi/2$ with θ_{hip}. During walking θ_{kne} ranges from $\theta_{min} = -60°$

to $\theta_{max} = 0°$[12], while during running θ_{kne} ranges from $\theta_{min} = -120°$ to $\theta_{max} = -15°$[13].

- During both walking and running the angle of the ankle θ_{ank} is in antiphase with θ_{kne}.During walking θ_{ank} ranges from $\theta_{min} = -5°$ to $\theta_{max} = 20°$[12], while during running θ_{ank} ranges from $\theta_{min} = -30°$ to $\theta_{max} = 25°$[13].

- During both walking and running the angle of the arm at the shoulder θ_{sho} is in counterphase with the angle of the same side leg at the hip θ_{hip}. During walking θ_{sho} ranges from $\theta_{min} = -5°$ to $\theta_{max} = 20°$, while during running θ_{sho} ranges from $\theta_{min} = -30°$ to $\theta_{max} = 0°$[11].

- During both walking and running the angle of the arm at the elbow θ_{elb} is in phase with θ_{sho}. During walking θ_{elb} ranges from $\theta_{min} = 5°$ to $\theta_{max} = 25°$, while during running θ_{elb} ranges from $\theta_{min} = 85°$ to $\theta_{max} = 110°$[11].

- All the angles formed by limbs on one side of the body are in counterphase with the angles formed by the corresponding limbs on the opposite side of the body.

- During walking the torso stands straight with the z axis, while during running it forms a constant angle of 8°[14].

Fig. 5 shows the time evolution of the angles θ_j over a period T for walking and Fig. 6 reports the same quantities for running. Both figures include a visualization of snapshots of our model during a period. The numerical details of both our walking and running bodies are provided in GitHub [15].

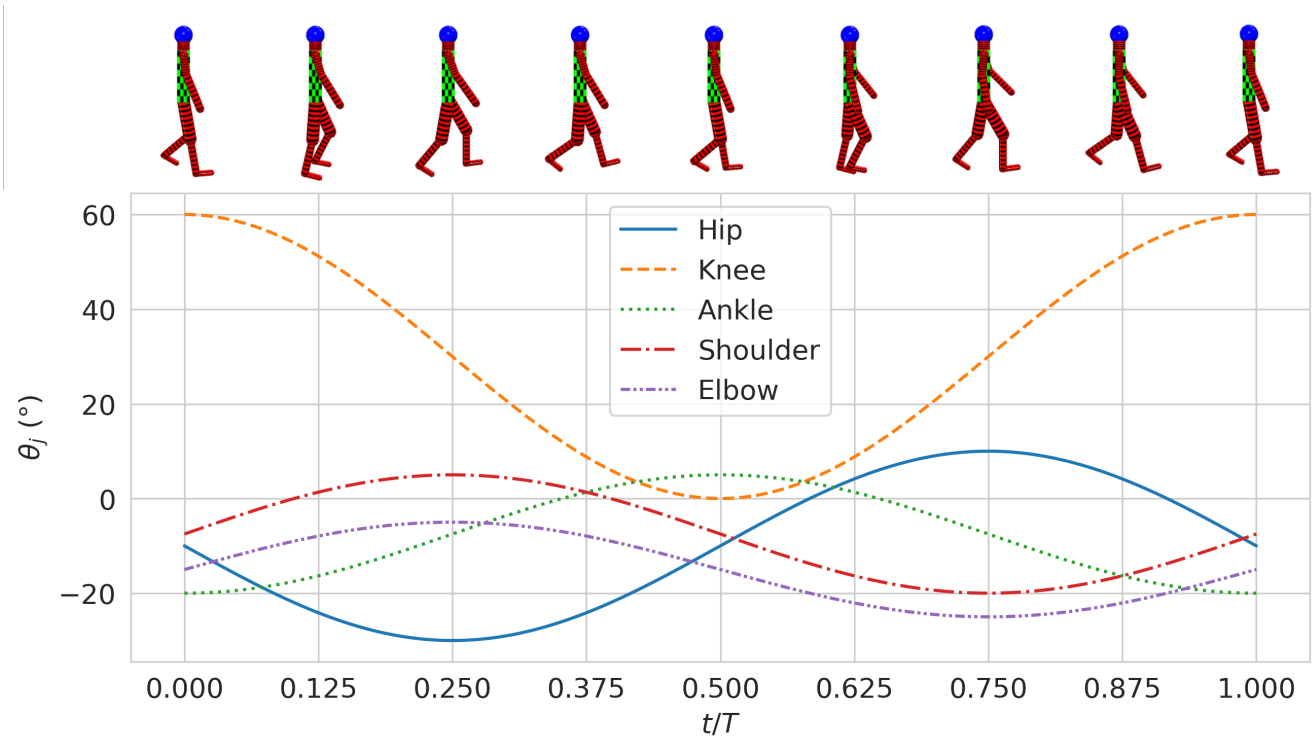

Figure 5: Angles θ_j of the model parts representing individual limbs on the right side of the body during walking for one period T. Upper images: illustrative snapshots of the resulting human walking model at intervals $T/8$.

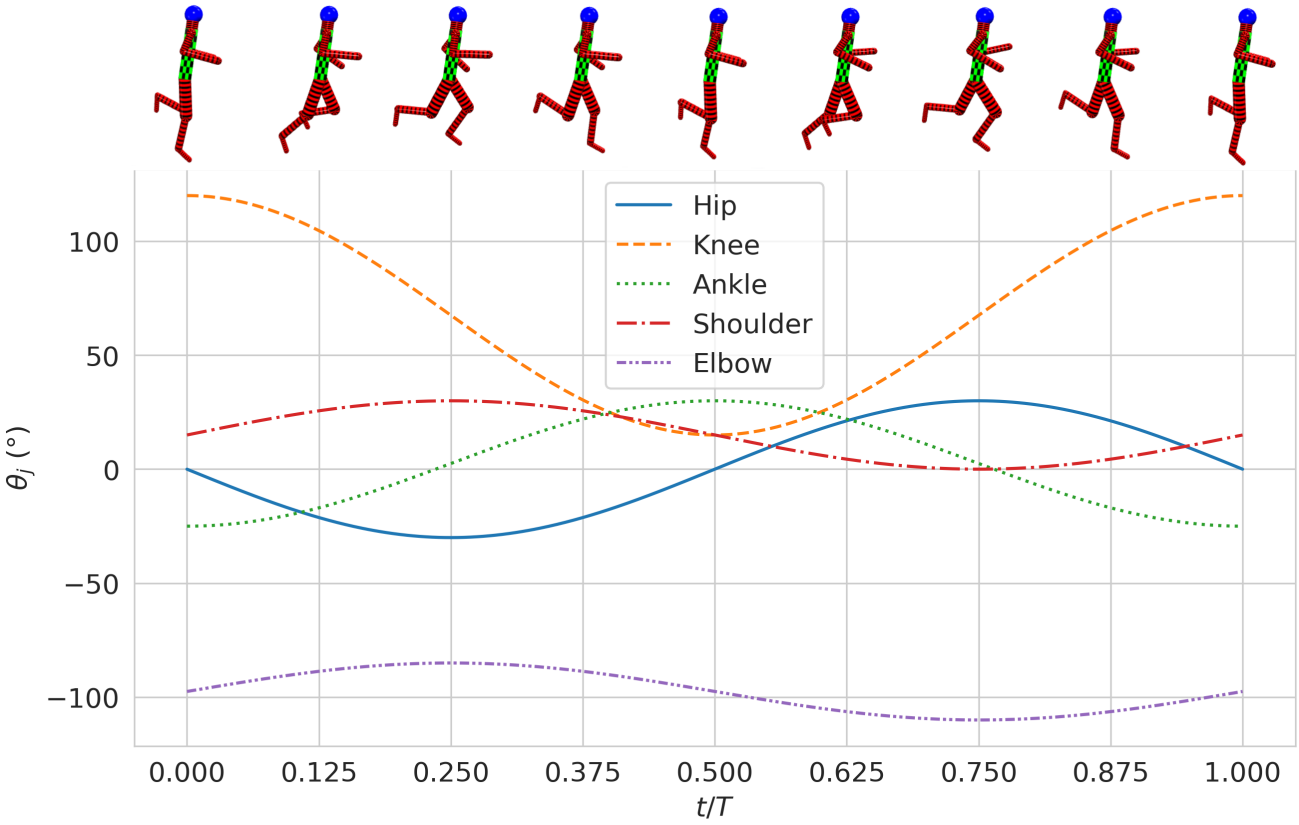

Figure 6: Same as Fig. 5 but for running instead of walking.

3 Technical implementation

The full code is available on GitHub [15]. We wrote the main code for the evaluation of the numerical results in C++. For data analysis and visualization we use Python. This section focuses on the C++ main code. We shortly describe the general organization of the code before delving into more details. As stated before the main objective is evaluating R_b for a complex body composed of base bodies. For this purpose, we need to evaluate the projection of a complex body on an arbitrary plane. To do so we generate a portion of said plane, project the elementary parts forming the complex body and evaluate the area of the total projection making sure not to double count overlapping projections. We then proceed to describe the implementation of the dynamics of the connected elementary bodies, which are employed to simulate a dynamic model of a walking/running human being. Note that in the context of the code, unless specified otherwise, when mentioning vectors we refer to `std::vector<double>` data structures, and when mentioning matrices we refer to `std::vector<vector<double>>` data structures.

3.1 Projection surface

The general idea is to generate a large number of straight lines parallel to $\mathbf{v}_{rel}$. These lines, which we call rays, are uniformly distributed in space. To evaluate the projected area, we count how many of these rays intersect the body whose projection we measure. A single ray can be parameterized as follows:

$$\mathbf{R}(\tau) = \mathbf{R}_0 + \tau \mathbf{v}_{rel}, \quad \tau \in \mathbb{R}. \tag{30}$$

We call $\mathbf{R}_0$ the "ray origin" of the specific ray. These rays are represented in the code by the `Ray` class. Its data members are a vector `R0` and a bool `Active`. `R0` contains the ray origin $\mathbf{R}_0$, and `Active` is a flag describing whether a ray has yet to hit a body section. `Ray` also contains a vector `V` as a static data member: `V` represents $\mathbf{v}_{rel}$, so its value is the same for each ray. The `Ray` methods `On` and `Off` allow to set `Active` to be `true` and `false` respectively, and the method `IsOn` returns the value of `Active`.

We generate the ray origins on a square lattice laying on the projection plane perpendicular to $\mathbf{v}_{rel}$. The side of the square dx determines the precision of our estimation. Since we want our code to be as efficient as possible we generate the ray origins only on a portion of the plane where the projection of the body could fall. To do so for each body whose projection we want to measure we consider a parallelepiped with sides parallel to the coordinate axes and sufficiently large to fully contain the entire body. We call this parallelepiped the simulation box. The ray origins are uniformly placed inside this projected simulation box. As the body is contained in the simulation

box, its projection will be contained in the projection of the simulation box. For each body we consider a different size simulation box, since we want it to be as tight around the body as possible to minimize the number or rays generated. When considering a dynamic body the simulation box has to contain the body at all times, as the rays are generated only once, not each time said body moves.

We implement this with the class `ProjSurf`. `ProjSurf` has as data members a vector of `Ray` objects `rays`, a double `dx` and a matrix `H`. As shown in Sec. 2.1.2 the projection of a parallelepiped is an irregular hexagon. `H[i]`, with `i` $= 1, 2, 3, 4, 5, 6$, contains the coordinates of the 6 vertices of the hexagonal projection of the box, while `H[0]` contains the coordinates of a vertex of the box that lays inside the projection surface. The `ProjSurf` constructor takes as arguments a vector `box`, a vector `vel` and a double `Dx`. `box` represents the lengths of the three sides of the simulation box, `vel` is $\mathbf{v}_{rel}$ and `Dx` is dx. The constructor first uses the function `FindMiddle` to find the vertex of the box shared by the faces of the box that would get wet, and assigns that vertex to `H[0]`. Then it calls the function `FindHexProj` to find the projection of the vertices of the "wet faces" onto a plane perpendicular to `vel` and passing through `H[0]`, and assigns the points found this way to the remaining elements of `H`.

Now that we have our hexagonal projection `H`, we only need to fill it with the ray origins. To do so we consider two perpendicular unit vectors laying on our projection plane, `u1` and `u2`. As `u1` we use the longer vector between `H[5]`$-$`H[0]` and `H[3]`$-$`H[0]`. Since we want `u1` to be a unit vector we normalize it after choosing it. We can not just take `H[i]` $-$ `H[0]` with a fixed `i` this because if the faces of which `H[i]` is a vertex are degenerate then `H[i]` $-$ `H[0]` $= 0$. `H[5]` is a vertex of two faces and `H[3]` is a vertex of a third one two vertices that do not share a face, so even if two of the three faces are degenerate at least one between `H[3]` $-$ `H[0]` and `H[5]` $-$ `H[0]` is a nonzero vector. `u2` is then derived by normalizing the cross product between `u1` and `vel`, ensuring that it is perpendicular to both. Then we find `maxu1` and `maxu2`, which are the maximum distance of `H[0]` and the projection along respectively `u1` and `u2` of `H[i]`. This way we can safely say that the hexagon defined by `H` is contained in the rectangle centered on `H[0]` with sides `2*maxu1*u1` and `2*maxu2*u2`. We generate points on a square lattice with side dx along `u1` and `u2` inside said rectangle, and we check if they are inside the hexagon we use the `Ray` constructor to build a `Ray` with that point as ray origin `R0` and we add it to `rays`. To check whether the points lay inside the hexagon we use the bool function `PointIsInsideT`. This function considers the triangles with vertices `H[0]`, `H[i]` and `H[i+1]`, with `H[i+1]` = `H[1]` when $i = 6$; these 6 triangles perfectly cover the hexagon. Then the function uses barycentric coordinates to determine if the point is inside each triangle, as shown in section 3.2 of Ref. [16]. If the function finds the point to be inside any of the triangles it returns `true`, otherwise it returns `false`. Having generated the ray origins we set the direction of the rays with `Ray::V = vel`.

The other main method of ProjSurf is the function BodyProj, which takes as argument a Body class object and returns the surface of its projection on the projection plane. The details of the Body class along with its methods are explored in Secs. 3.2 and 3.3. BodyProj counts how many of the rays elements intersects the body by calling the Body method Check on each Ray, and it returns the number of hitting rays times dx squared, which corresponds to the area of a single cell of the square lattice of ray origins. We can also call BodyProj with as optional arguments two doubles tmin and tmax, and an unsigned int nstep to compute the S_b of a dynamic body according to Eq. (6). To do so we use the Body method Move, which when given as argument a double t moves the body to its configuration at time t. BodyProj moves the Body at nstep time steps between tmin and tmax, which will typically be the extremes of one period of motion of the Body, and at each time step counts the number of rays that intersect the Body, before returning the nstep-average number of hitting rays times dx squared. We call the quantity $dt = T/\texttt{nstep}$ the time step.

3.2 Static bodies

We define a parent class Body from which all the classes representing the various shapes inherit. The Body methods that we discuss first are Prime, Check and Anal. Each derived class has its own implementation of these methods that overrides the original one, but in general Prime prepares the Body to be checked, Check takes as argument a Ray and returns whether it intersects the Body, and Anal takes as argument a vector representing $\mathbf{v}_{rel}$ and a double representing v_b, and returns the analytical solution R_b. We divide the checking process into Prime and Check because an efficient way to check if a Ray from a ProjSurf intersects a base body is to project said body on the projection surface and then check if the R0 of the ray is inside the projection of the body. Since we have to check a large number of rays it is then more efficient to first project the body by calling Prime once, and then call Check for each ray without having to project the body each time. Check also checks that the ray is active using the Ray method IsOn, and turns off the ray using the Off method if it finds that it intersects the body, thus solving the issue of different elementary parts shadowing each other. The function Prime groups all preliminary operations which are independent of the specific Ray considered.

Sphere We implement spheres through the Sphere class, derived from Body. Its additional data members are a vector cent representing the position of its center, a double rad representing its radius and a vector Hcent representing the projection of cent on a ProjSurf. As shown in Sec. 2.1.1 the projection of a sphere on a plane is a disk whose center is the projection of the center of the sphere on the plane and

with the same radius as the sphere. `Prime` simply finds the projection of `cent` onto the projection surface and assigns it to `Hcent`. When called on a `Ray`, `Check` returns `true` if the distance between `R0` and `Hcent` is less than or equal to `rad`, and `false` otherwise. `Anal` returns the analytical solution of R_s, Eq. (12).

Parallelepiped We implement parallelepipeds with the `Parallelepiped` class, which inherits from `Body`. Its additional data members are a vector `cent` representing the position of its center, a matrix `side`, where `side[i]` contains the side $\mathbf{s}_i$, and a matrix `H` analogous to the `H` data member of `ProjSurf`. `Prime` projects the vertices of the wet faces of the parallelepiped onto the projection surface, setting `H[0]` equal to the projection of the vertex shared by all the seen faces, and `H[i]` equal to the vertices of the hexagonal projection. `Check` then uses `PointIsInsideT` to check if `R0` is inside the hexagon defined by `H`. `Anal` returns the analytical solution of R_p, Eq. (18).

Capsule We implement capsules with the `Capsule` class, derived from `Body`. Its additional data members are two vectors, `l1` and `l1` representing the position of the two endpoints of its axis, a double `rad` representing its radius and two vectors, `H1` and `H2` representing the projection of `l1` and `l2` on a `ProjSurf`. As shown in Sec. 2.1.3 the projection of a capsule on a plane is a stadium whose axis is the projection of the axis of the capsule and with the same radius as the capsule. `Prime` computes the projection of `l1` and `l2` onto a projection surface and assigns them to `H1` and `H2`. When called on a `Ray`, `Check` uses the function `PointSegDist` to calculate the distance between `R0` and the segment defined by the two endpoints `H1` and `H2`, and then returns true if it is less or equal to `rad`, and `false` otherwise. `Anal` returns the analytical solution of R_c, Eq. (27).

Composite object We implement a complex body formed of several elementary bodies by means of the `ManyBody` class, also derived from `Body`. Its additional data members are a vector of pointers to `Body` objects called `bodies`. Calling the `Prime` method has the effect of calling `Prime` with the same arguments on all the `Body` objects contained in `bodies`, preparing all of them to be checked. Similarly when `Check` is called on a `Ray`, it calls `Check` for all the elements of `bodies` and returns `true` if the `Ray` intersects any of them.

3.3 Body dynamics

In this section we describe our implementation of the relative movements of the individual body element in `ManyBody`. We need our elementary bodies to move not just independently but also relative to each other. We accomplish this with the `Body` data

member SubBodies, which is a vector of pointers to Body objects. If we have a Body A, and the SubBodies of A contains a pointer to another Body B, we say that B is a sub-body of A, and A is the super-body of B. A sub-body moves in the system of reference in which its super body is still. As a result, when so if the super-body moves so do its sub-bodies. For example in both our walking and running man the Capsule representing the forearm is a sub-body of the Capsule representing the upper arm, so when the upper arm swings back and forth this movement propagates to the forearm.

We model the movement of each base body as a translation and a rotation around an axis. The center of and axis of rotation of each Body is contained in the vector data members rotcent and rotax respectively. We call $\mathbf{T_r}(t)$ and $\Theta(t)$ respectively the translation and the angle of the rotation that move the body from its original position to its position at time t. Since we want our movements to be periodic we write both $\Theta(t)$ and $\mathbf{T_r}(t)$ as a Fourier series with a finite amount of terms. These terms are contained in the Body data members trans and w, which are a vector and a matrix respectively. The even elements of trans and w represent the sin terms of the expansion, and the odd elements represent the cos terms. $\mathbf{T_r}(t)$ is then defined as:

$$\begin{aligned}\mathbf{T_r}(t) = &\sum_{\mathtt{i}=0}^{i_{max}} \mathtt{trans[2*i]} \sin\left(\frac{t}{T}\frac{2\pi}{\mathtt{i}+1}\right) + \sum_{\mathtt{j}=0}^{j_{max}} \mathtt{trans[2*j+1]} \cos\left(\frac{t}{T}\frac{2\pi}{\mathtt{j}+1}\right), \\ &\text{with} \quad i_{max} = \frac{\mathtt{trans.size()}}{2} - 1, \quad j_{max} = \frac{\mathtt{trans.size()}}{2} - 2.\end{aligned} \tag{31}$$

$\Theta(t)$ is similarly defined as:

$$\begin{aligned}\Theta(t) = &\sum_{\mathtt{i}=0}^{i_{max}} \mathtt{w[2*i]} \sin\left(\frac{t}{T}\frac{2\pi}{\mathtt{i}+1}\right) + \sum_{\mathtt{j}=0}^{j_{max}} \mathtt{w[2*j+1]} \cos\left(\frac{t}{T}\frac{2\pi}{\mathtt{j}+1}\right), \\ &\text{with} \quad i_{max} = \frac{\mathtt{w.size()}}{2} - 1, \quad j_{max} = \frac{\mathtt{w.size()}}{2} - 2.\end{aligned} \tag{32}$$

Our bodies also need to know their current position in time. This is accomplished with the Body data member t, which denotes at which point in time the body finds itself. t equals the time t expressed in units of T.

We now have all the elements needed to properly describe the Move method. While it changes slightly for each body the better part of it is identical. When called with as argument a double T it uses Eq. (31) calculates the transition step vector delta which translates the body from its current position at time t to its new position at time T: delta = $\mathbf{T_r}$(T) − $\mathbf{T_r}$(t). It then uses Eq. (32) to calculate the rotation angle theta used to rotate the body to the its position at time T: theta = $\mathbf{w}$(T) − $\mathbf{w}$(t). With theta and rot it uses the function RotMat to compute the rotation matrix rotmat which when applied to a vector rotates it around the origin around the axis rotax. Applying this matrix directly is sufficient to rotate vectors that represent the distance between two

points in space, such as the sides of a parallelepiped, while to rotate vectors describing absolute positions in space around `rotcent` we use the function `Rotate`.

`Move` then applies the translation and rotation to the body. In a sphere it translates and rotates its center, in a parallelepiped it translates and rotates the center and rotates the sides, in a capsule it rotates and translates the endpoints of the axis. `cent` is translated too, so that the center of rotation translates coherently with the rest of the body.

At last `Move` calls the `Body` method `BeMoved` on all its sub-bodies. `BeMoved` takes as input `trans`, `rotmat` and `rotcent` and moves the sub-body accordingly. This includes translating and rotating the `rotcent` and `rotax` of the sub-body, and rotating the vectors in its `trans`. This way the system of reference in which the sub-body moves is subject to the same rototranslation. This procedure works recursively, so that `BeMoved` is called on the sub-body's own sub-bodies (if any) ensuring that they are moved as well.

3.4 Code validation

We proceed to check that our code works properly. To do so we compare the numerical results we obtain for a sphere, parallelepiped and capsule with the analytic results of Sec. 2, which the code computes too. We call $\tilde{R}_b(dx)$ the numerical estimation of R_b evaluated with our code with a step dx. The error $\Delta R_b(dx)$ then is the difference between $\tilde{R}_b(dx)$ and the exact R_b evaluated analytically:

$$\Delta R_b(dx) = \tilde{R}_b(dx) - R_b. \tag{33}$$

The absolute deviation is then $|\Delta R_b(dx)|$. We assume that the absolute deviation follows a power law:

$$|\Delta R_b(dx)| = \alpha \, dx^{\beta}, \tag{34}$$

where α and β are parameters to determine. We check that this is the case. With a set body, $\mathbf{v}_r$ and v_b we evaluate ΔR_b for different dx values ranging from 0.0001 m to 0.2 m. We then fit the deviations to the power law. To do so we first take the logarithm on both sides of Eq. (34):

$$\ln(|\Delta R_b(dx)|) = \ln(\alpha) + \beta \ln(dx). \tag{35}$$

We then evaluate α and β by means of a least squares regression. We use the same $\mathbf{v}_r$ and v_b for each body to better be able to compare the results. We carry out this test for $v_{tail} = 0.5 \; v_{fall}$, $v_{cross} = 0.25 \; v_{fall}$ and $v_b = 2 \; v_{fall}$. As solids consider a sphere with radius $r = 0.5$ m, a parallelepiped with sides $\mathbf{s}_1 = (0.4, 0, 0)$ m, $\mathbf{s}_2 = (0, 0.6, 0)$ m, $\mathbf{s}_3 = (0, 0, 0.8)$ m, and a capsule with a radius $r = 0.3$ m and $\Delta\mathbf{l} = (0.4, 0.4, 0.4)$ m.

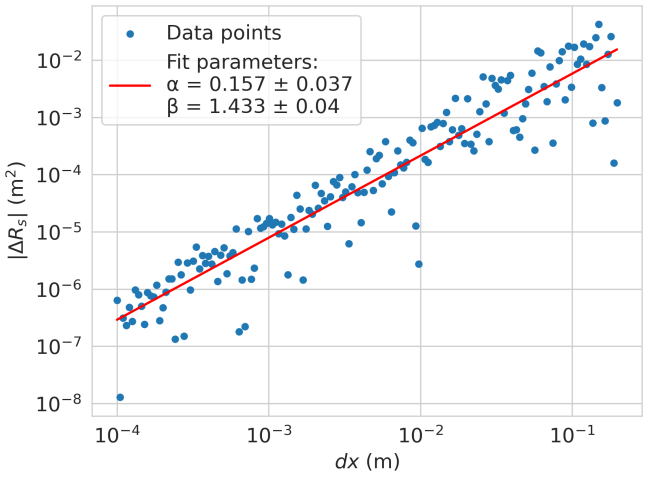

Figure 7: The absolute deviation $|\Delta R_s(dx)|$ as a function of the resolution dx of the array of probing rays, for a sphere with radius $r = 0.5$ m and velocities $v_{tail} = 0.5\ v_{fall}$, $v_{cross} = 0.25\ v_{fall}$ and $v_b = 2\ v_{fall}$. The red line represents the power law fit.

We can see the results for the sphere on Fig. 7, for the parallelepiped on Fig. 8 and for the capsule on Fig. 9. In all these cases we find that the power law of Eq. (34) provides a good fit for $|\Delta R_b(dx)|$. The most important difference is in the exponent β: for the parallelepiped we find $\beta = 1.062$ compared to $\beta = 1.447$ and $\beta = 1.537$ for the sphere and capsule, respectively. This slower convergence of $\tilde{R}_p(dx)$ to R_p could be due to the more complex shape of the projection of the parallelepiped, with sharp edges and vertices.

Figs. 10, 11 and 12 report a comparison of $R_b(v_b)$ and $\tilde{R}_b(v_b, dx)$ as a function of v_b for the three considered elementary solids. The parameters are as before and $dx = 0.001$ m. Clearly the precision is already good, and it is sufficient to determine the minima of $R_b(v_b)$. These graphs also provide examples of situations where the shape of the body determines the existence and value of an optimal speed.

Another aspect of the code we check is its ability to handle rain shadowing in composite bodies correctly. To do so we consider a complex body composed by two parallelograms: p_1 with sides $(0.8, 0, 0)$ m, $(0, 0.5, 0)$ m and $(0, 0, 0.8)$ m, and p_2 with sides $(0.6, 0, 0)$ m, $(0, 0.3, 0)$ m and $(0, 0, 0.6)$ m. We call $\mathbf{c}_1$ the center of p_1, and $\mathbf{c}_2$ the center of p_2. We consider $v_b = 0.52\ v_{fall}$, $v_{tail} = 0$ and $v_{cross} = 0.03\ v_{fall}$. We call the total wetness of this double parallelepiped R_{2p}. We evaluate R_{2p} as we move $\mathbf{c}_2$ along the y axis starting from $\mathbf{c}_2 = \mathbf{c}_1$. We expect that at the start p_2 is contained inside p_1, so $R_{2p} = R_{p1}$, but as $|\mathbf{c}_2 - \mathbf{c}_1|$ increases we expect R_{2p} to remain constant while p_2 remains fully inside p_1, then to increase as p_2 gradually moves out, and finally remain constant again as p_2 is fully outside. Furthermore since $\mathbf{v}_{rel}$ has no y component, $p1$

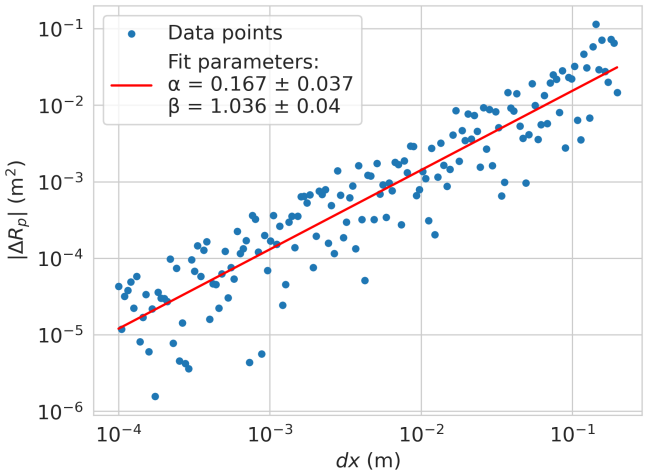

Figure 8: Same as Fig. 7 for a parallelepiped with sides $\mathbf{s}_1 = (0.4, 0, 0)$ m, $\mathbf{s}_2 = (0, 0.6, 0)$ m, $\mathbf{s}_1 = (0, 0, 0.8)$ m.

and $p2$ do not overshadow one another as long as they are no longer in contact, so we expect the final value of R_{2p} to simply be $R_{p1} + R_{p2}$. The numerical results reported in Fig. 13 confirms that R_{2p} behaves exactly as expected.

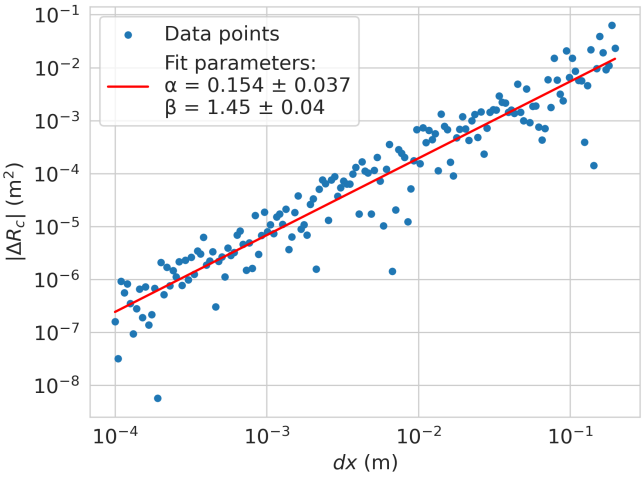

Figure 9: Same as Fig. 7 for a capsule with a radius $r = 0.3$ m and $\Delta\mathbf{l} = (0.4, 0.4, 0.4)$ m.

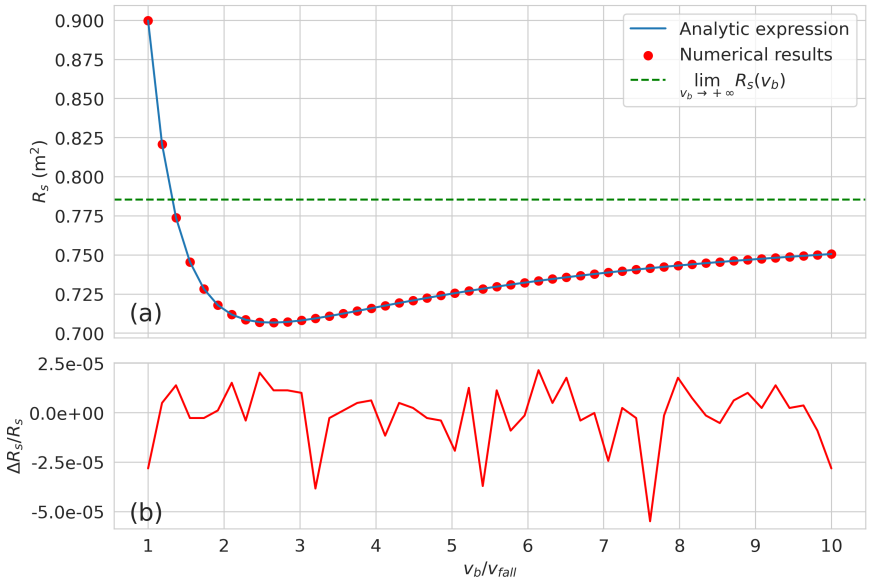

Figure 10: $R_s(v_b)$ and $\tilde{R}_s(v_b, dx)$ of a sphere for a variable v_b and a fixed $dx = 0.001$ m. (a) Comparison of the exact analytic wetness $R_s(dx)$ (line) and the numerical estimation of the same quantity $\tilde{R}_s(dx)$ (dots) obtained for a sphere of radius $r = 0.5$ m with a rain resolution $dx = 0.001$ m. The velocities are $v_{tail} = 0.5\ v_{fall}$, $v_{cross} = 0.25\ v_{fall}$ and $v_b = 2\ v_{fall}$. (b) The relative deviation $\Delta R_b(dx)$ of the numerical and exact estimates.

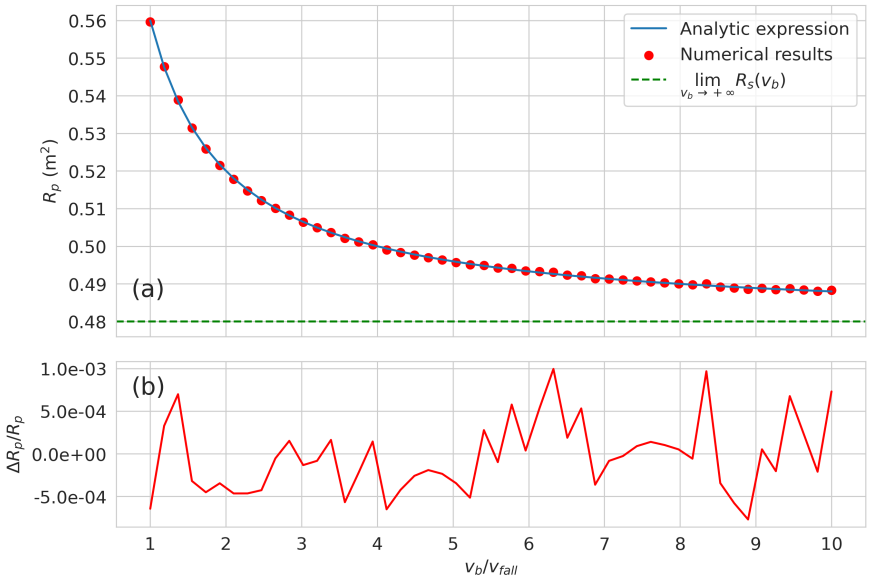

Figure 11: Same as Fig. 10, but for a parallelepiped with sides $\mathbf{s}_1 = (0.4, 0, 0)$ m, $\mathbf{s}_2 = (0, 0.6, 0)$ m and $\mathbf{s}_3 = (0, 0, 0.8)$ m.

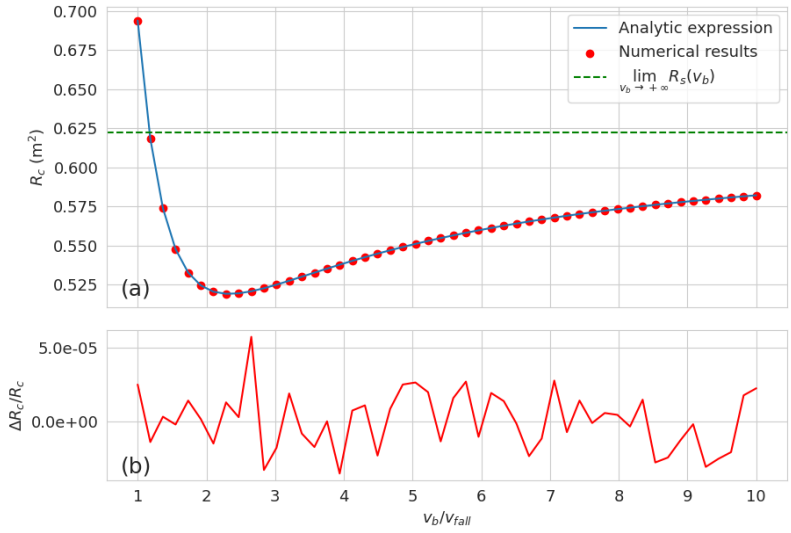

Figure 12: Same as Fig. 10, but for a capsule with a radius $r = 0.3$ m and $\Delta\mathbf{l} = (0.4, 0.4, 0.4)$ m.

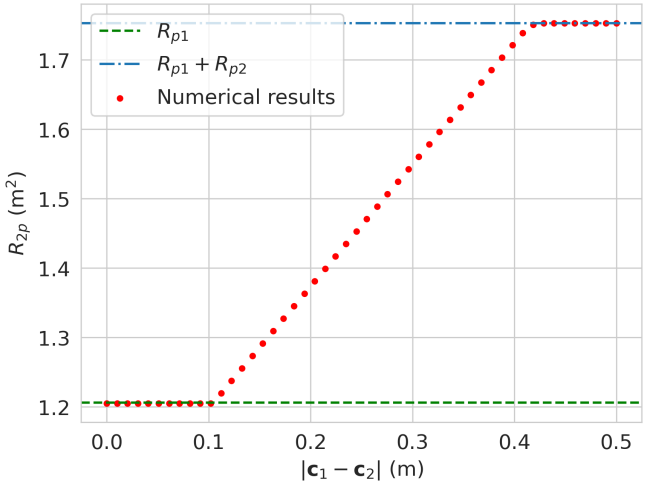

Figure 13: Wetness R_{2p} of a complex body composed by two parallelepipeds p_1 and p_2 as the distance between their centers $|\mathbf{c}_2 - \mathbf{c}_1|$ increases, $dx = 0.001$ m.

4 Results

4.1 Error analysis

To estimate the error on the numerical estimation of the wetness $\tilde{R}_b$ of the walking and running bodies we face additional challenges compared to the elementary bodies considered in Sec. 4.1. First of all we no longer have access to an analytic solution with which to compare the numerical results. Second we now have two sources of discretization error: the finite number of rays and the finite number of time steps.

We assume the two errors to be independent of each other, so we can evaluate them separately. We assume that both the discretization errors follow an asymptotic power law valid for small dx and dt. We can then write $\tilde{R}_b(dx, dt)$ as follows:

$$\begin{aligned} \tilde{R}_b(dx, dt) &= R_b + E_{dx}(dx) + E_{dt}(dt), \\ E_{dx}(dx) &\simeq A_{dx}\, dx^{B_{dx}}, \\ E_{dt}(dt) &\simeq A_{dt}\, dt^{B_{dt}}, \end{aligned} \tag{36}$$

where E_{dx} and E_{dt} are the discretization errors functions of dx and dt respectively. We choose a fixed value of dx and evaluate $\tilde{R}_b$ for different values of dt, then use a least squares regression to fit the data and find an estimate of $R_b + E_{dx}$ and its error. Once we have this estimate we can evaluate the absolute deviation similarly as we did in Sec. 3.4:

$$|\Delta R_b(dt)| = \left|\tilde{R}_b(dx, dt) - (R_b + E_{dx})\right|. \tag{37}$$

We can then fit this absolute deviation with a power law as done before:

$$|\Delta R_b(dt)| = \alpha_{dt}\, dx^{\beta_{dt}} \tag{38}$$

To study the error due to dx we do the same thing but with dt and dx switched.

We carry out this test with the same $\mathbf{v}_{rel}$ and v_b as in Sec. 3.4: $v_{tail} = 0.5\, v_{fall}$, $v_{cross} = 0.25\, v_{fall}$ and $v_b = 2\, v_{fall}$. We carry out the error analysis on dx with a fixed of $dt = 0.02\, T$, the error analysis on dt with a fixed $dx = 0.001$ m. Figs. 14 and 15 report the results of the error analysis on dt for the walking and running bodies respectively. Figs. 16 and 17 report the results for the error analysis on dx for the walking and running bodies respectively. The error due to dx on the walking and running bodies behaves similarly to the one on the three elementary solids studied in Sec. 3.4. The error due to dt exhibits both a faster convergence to zero and a smaller prefactor α compared to the one due to dx. Furthermore, almost all of the computational complexity of the code is due to generating and checking rays, and the number of rays generated and checked is proportional to dx^{-2}, while the number of times they are checked is proportional to dt^{-1}, thus makes it more computationally efficient to decrease dt than dx. The error due to dt is then much computationally cheaper to reduce than the error due to dx.

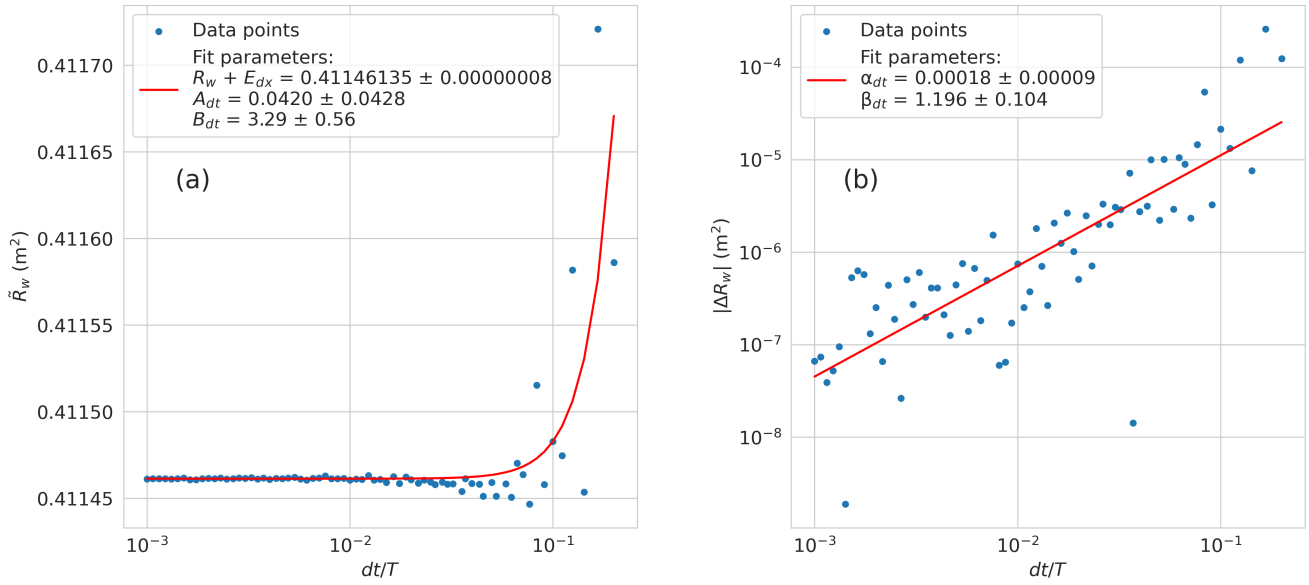

Figure 14: Analysis on the error due to finite dt for the walking body. (a) Estimated wetness $\tilde{R}_w$ as a function of the time step dt; the red line represents the power law fit of Eq. (36). (b) Absolute deviation $|\Delta R_w(dt)|$ as a function of the time step dt; the red line represents the power law fit of Eq. (38). In these calculations we fix $dx = 0.001$ m, $v_{tail} = 0.5\ v_{fall}$, $v_{cross} = 0.25\ v_{fall}$ and $v_b = 2\ v_{fall}$.

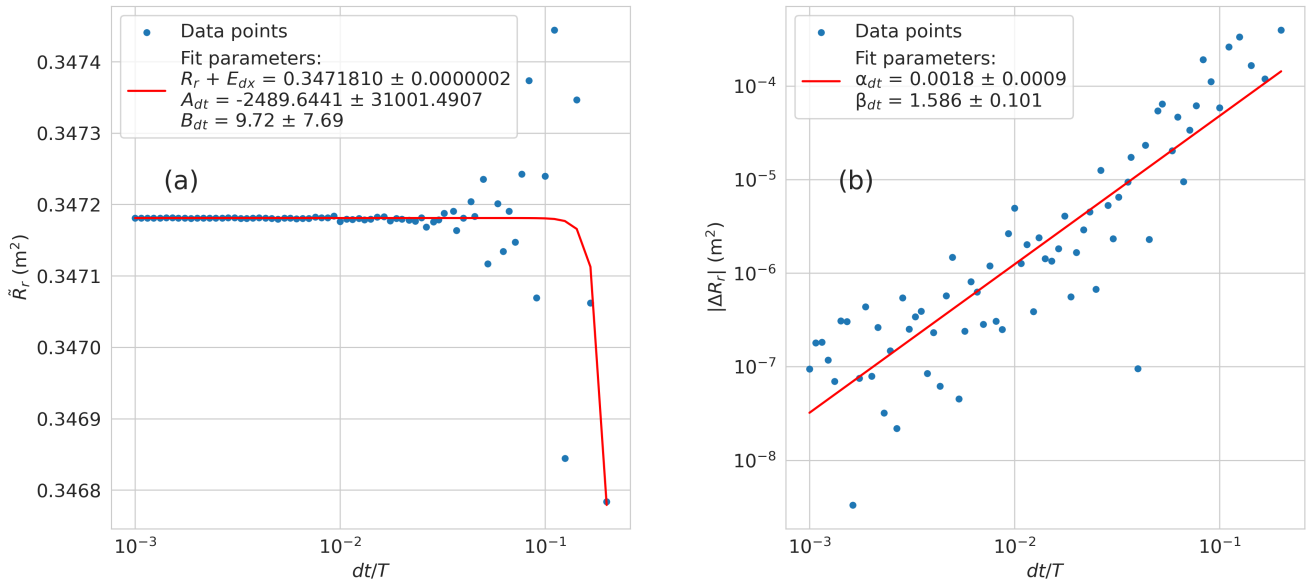

Figure 15: Same as Fig. 14 but for the running body.

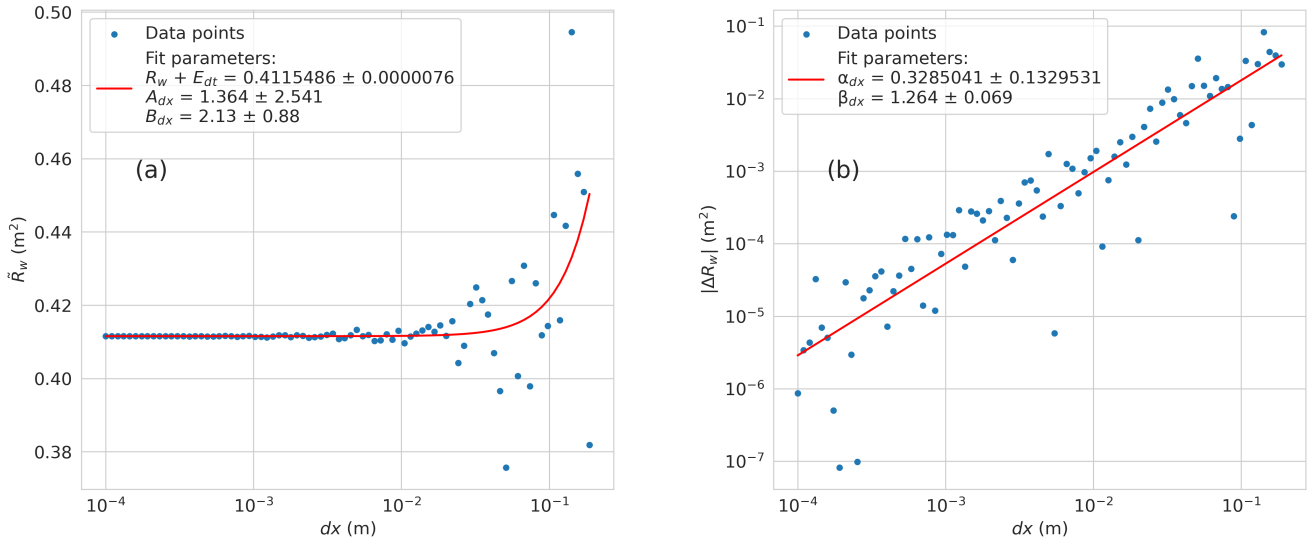

Figure 16: Same as Fig. 14 but for varying dx and fixed $dt = 0.02\,T$.

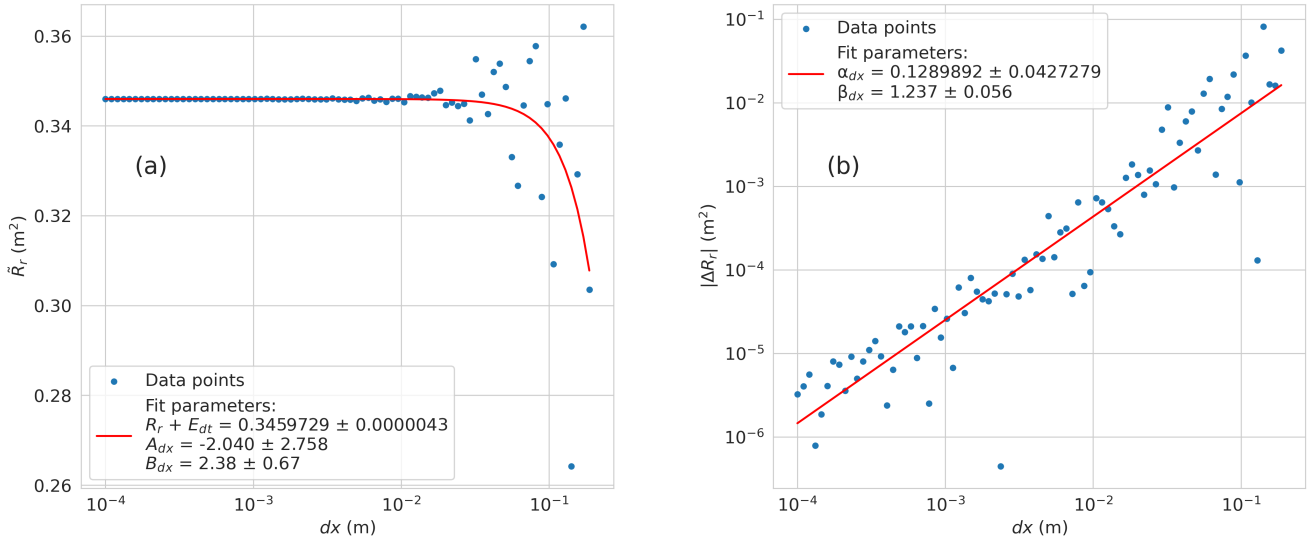

Figure 17: Same as Fig. 16 but for the running body.

4.2 Optimal velocity

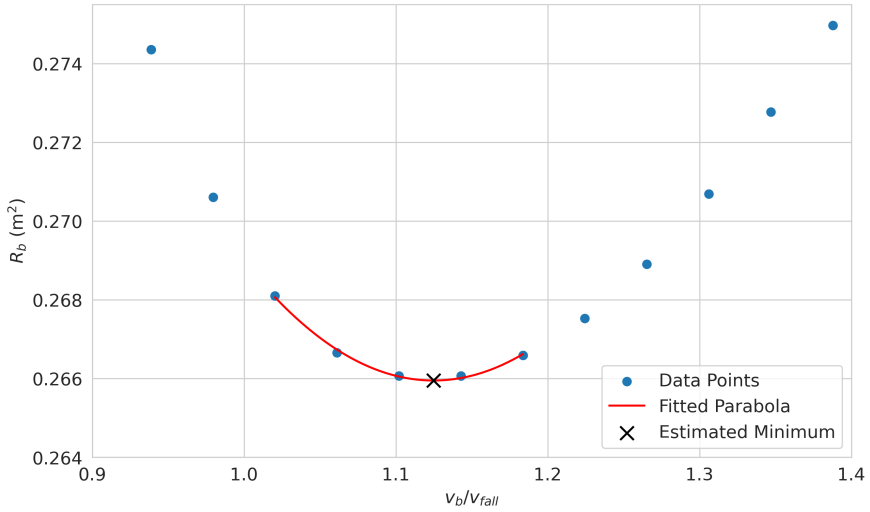

Figure 18: Example of estimation of v_{opt} for a running body found in the range $[0, 2\ v_{fall}]$ with the following conditions: $v_{tail} = 0.75\ v_{fall}$, $v_{cross} = 0.5\ v_{fall}$, $dx = 0.001$ m, $dt = 0.1\ T$, $N_v = 50$, $N_{fit} = 5$. The red line represents the parabola fit, and the black cross represents the minimum of v_{opt} estimated by the fitted curve.

We now study the v_{opt} of the walking and running bodies under varying values of v_{tail} and v_{cross}. We first define a range of speeds $0 \leq v_b \leq v_{max}$ we deem acceptable for running and walking. v_{max} represents the maximum velocity a body can achieve. Since we measure velocities in units of v_{fall} we first need to know its value. Rain falls at different speeds depending on the size of the raindrops, but the minimum speed for precipitations is found around 2.5 m/s [17]. As an upper bound for the running speed we consider the average speed of an elite marathon runner, which is around 5 m/s [18], corresponding to 2 v_{fall}. Furthermore a fast walking speed for healthy adults is found to be around 1.75 m/s [19], corresponding to 0.7 v_{fall}.

We approximate the shape of $R_b(v_b)$ around its minima with a parabola $P(v_b)$ defined as follows:

$$P(v_b) = R_{min} + k(v_b - v_{opt})^2. \tag{39}$$

For each value of v_{tail} and v_{cross} taken into consideration, we evaluate R_b at N_v values of v_b in the range $[0,\ v_{max}]$ and we take as an initial estimate of v_{opt} the value corresponding to the minimum R_b from those evaluated. We consider N_{fit} values of v_b centered around this initial estimate and the respective $R_b(v_b)$ values, and fit these data points on Eq. (39) by means of a least squares regression. The N_{fit} values of v_b we consider are spaced as the original N_v values, allowing us to use values of v_b for

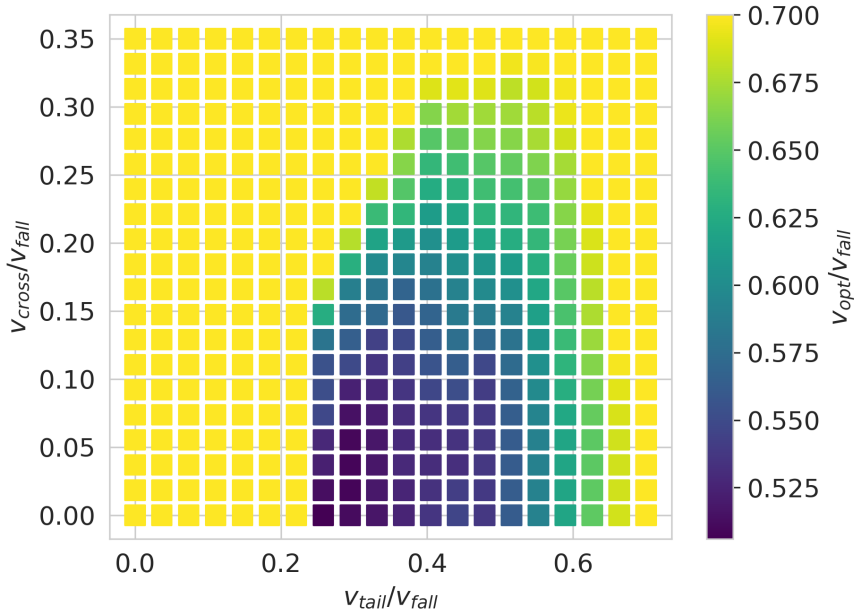

Figure 19: Numerically evalued v_{opt} of the walking body as a function of v_{tail} and v_{cross}. $dx = 0.001$ m and $dt = 0.1\ T$.

which we have already calculated R_b. This fit provides an estimation of v_{opt} and R_{min} along with their relative error. If the fit produces values of v_{opt} outside of the range $[0,\ v_{max}]$, or fails due to the minimum not existing or being much larger than v_{max}, we estimate $v_{opt} = v_{max}$ and $R_{min} = R_b(v_{max})$. We show an example of this approach in Fig. 18.

Whenever we evaluate v_{opt} and R_{min} in this chapter we do so with $dx = 0.001$ m, $dt = 0.1\ T$, $N_v = 50$ and $N_{fit} = 5$ for both the walking and the running body. Figs. 19 and 20visualize v_{opt} as a function of v_{tail} and v_{cross}, as color maps for the walking and running body respectively. We compare these results with the v_{opt} of a sphere and a parallelepiped with sides $\mathbf{s}_1 = (0.14, 0, 0)$ m, $\mathbf{s}_2 = (0, 0.42, 0)$ m, $\mathbf{s}_1 = (0, 0, 1.58)$ m evaluated analytically, shown in Figs. 21 and 22.

As we can see the v_{opt} of the walking and running bodies presents major quantitative differences from the v_opt of the sphere and the parallelepiped. For the sphere the minimum value of v_{opt} is 2 v_{fall}, which at the end of the whole range of v_{opt} considered for the walking and running body: if a sphere were an accurate model of a human body the solution would always be to move as fast as possible. We see that our more detailed modeling of the human body shows this to be false, as v_{opt} assumes values well below 2 v_{fall} over a good range of values of v_{tail} and v_{cross}. The modeling of the human body as a parallelepiped predicted that if v_{opt} exists then its value equals that of v_{tail}; this prediction is also contradicted more realistic model that for both walking and running predicts a value of v_{opt} exceeding v_{tail}, and decreasing as v_{fall} increases. One common

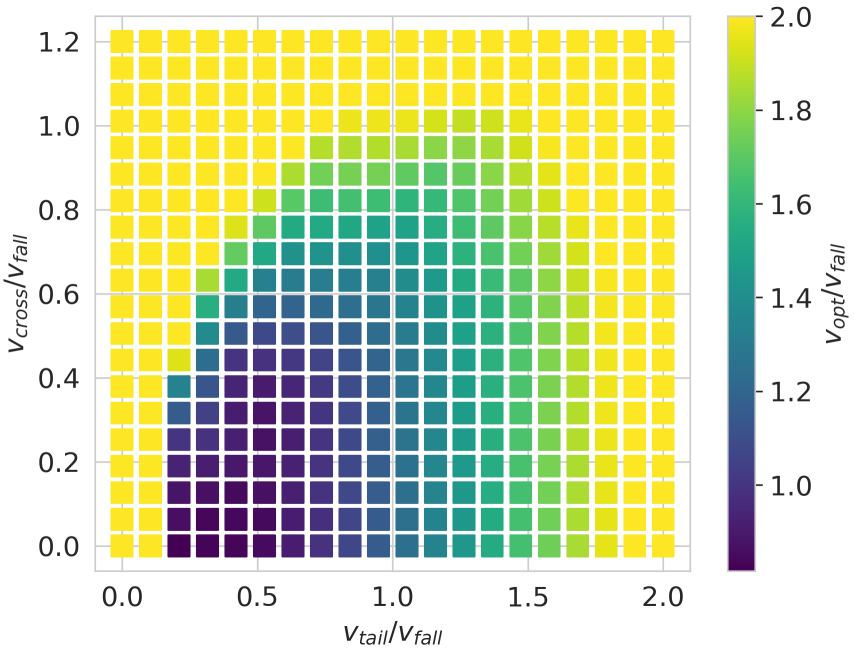

Figure 20: Same as Fig. 19 but for the running body.

property of v_{opt} shared by all the considered body models is the fact that for v_{opt} to be finite v_{tail} needs to be positive. While it is possible that finite v_{opt} exist for $v_{tail} \leq 0$ in the walking and running bodies, none was found in the $[0, v_{max}]$ range in either model. Comparing the results of the walking and running bodies we note that the v_{opt} of the walking body, which ranges from 0.52 v_{fall} to 0.70 v_{fall} generally have lower values than the v_{opt} found for the running body, which range from 0.9 v_{fall} to 2 v_{fall}.

We now ask ourselves if it is better to walk or run in the rain. To answer this question we compare the values of the minimum wetness R_{min} evaluated for the walking and running body with the same values of v_{tail} and v_{fall}, and check which body achieves the smaller one. We obtain a comparison shown in Fig. 23. In the vast majority of horizontal wind velocity components the best course of action is to run, either as fast as possible or at v_{opt}, but there is a small but important subset of wind velocities in which walking is preferable: $v_{tail} \approx 0.6\, v_{fall}$ and $v_{cross} \leq 0.2\, v_{fall}$.

We now illustrate the detailed behavior of v_{opt} as a function of v_{tail} for fixed values of v_{cross}. For a finite number of v_{cross} values we search for v_{opt} as before for multiple values of v_{tail} and consider only the results in the range $0 \leq v_{opt} \leq v_{max}$. Figs. 24 and 24 report these detailed v_{opt} curves for the walking and running bodies respectively. The presence of v_{cross} systematically increases v_{opt}. A sufficiently large v_{opt} can even eliminate v_{opt} completely, although the effect diminishes as v_{tail} increases. Furthermore, the detailed plots confirm that the walking body achieves lower values of v_{opt} than the running body. This is mainly due to the parallelepiped representing the

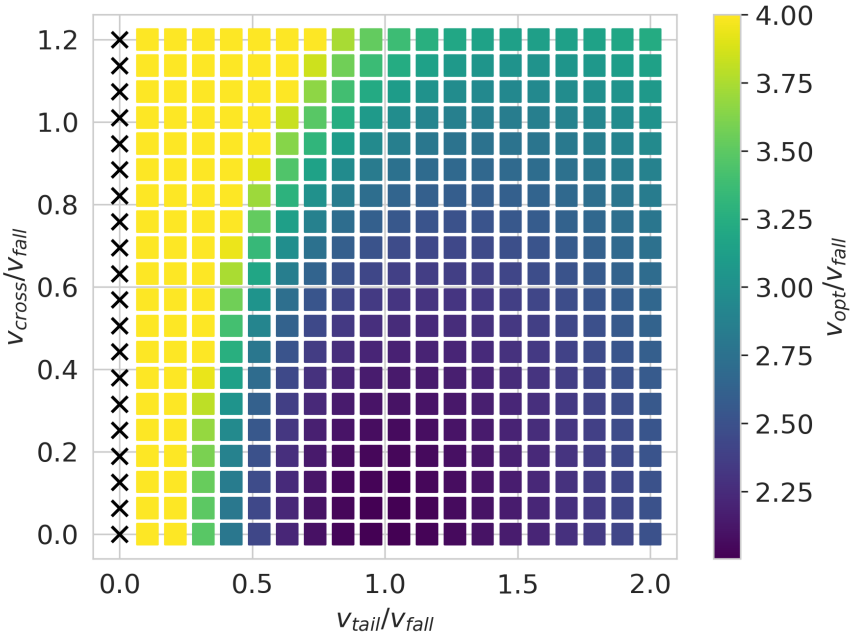

Figure 21: v_{opt} of a sphere as a function of v_{tail} and v_{cross} evaluated analytically with Eq. (14). The black crosses indicate that no v_{opt} exists for those values of v_{tail} and v_{cross}.

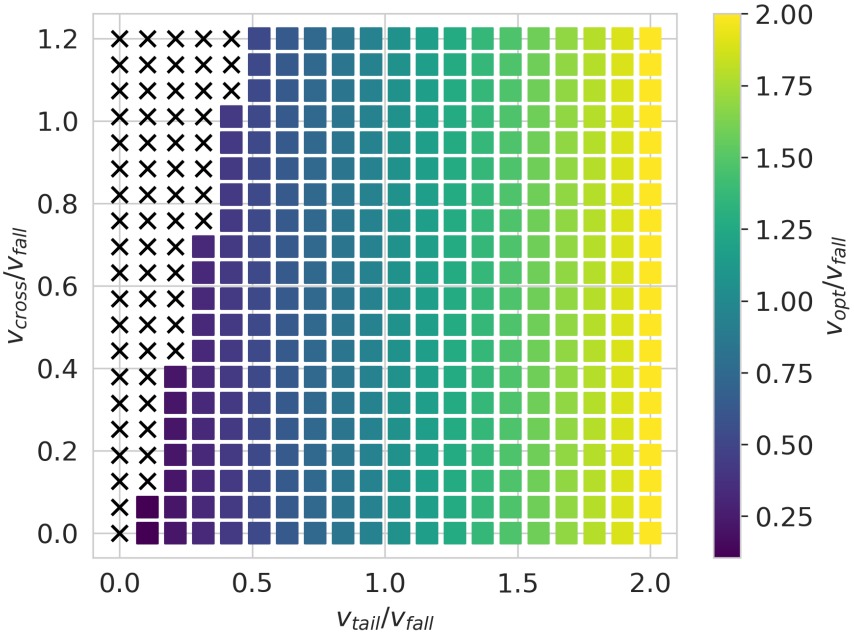

Figure 22: Same as Fig. 21 but for a parallelepiped with sides $\mathbf{s}_1 = (0.14, 0, 0)$ m, $\mathbf{s}_2 = (0, 0.42, 0)$ m and $\mathbf{s}_3 = (0, 0, 1.58)$ m. v_{opt} evaluated analytically with Eq. (20).

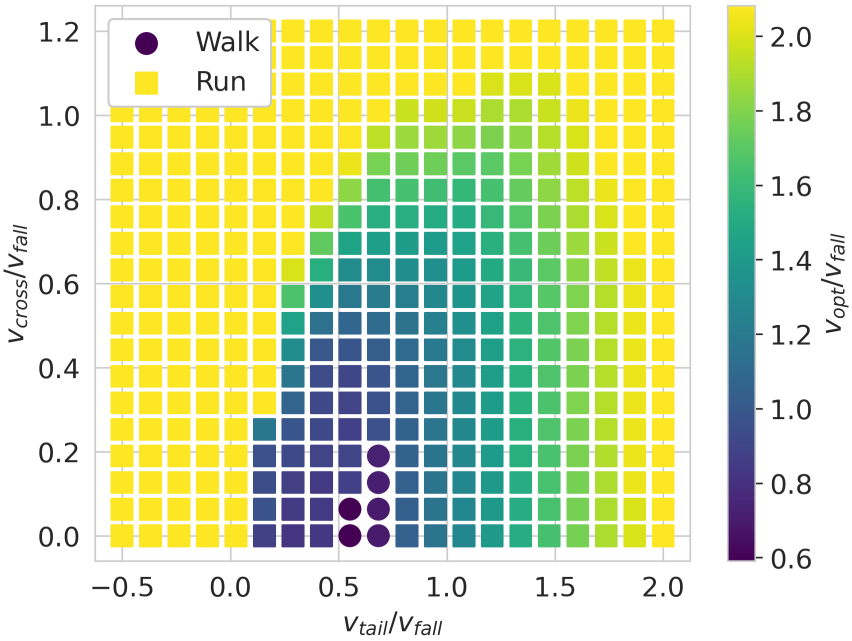

Figure 23: v_{opt} of the body that achieves the least wetness between the walking and running bodies as a function of v_{tail} and v_{cross}.

torso acquiring a forward angle in running. This inclination penalizes the low-end of speeds. In both graphs we observe discontinuities at smaller values of v_{tail}, with sharp increases in the value of v_{opt} as v_{tail} is reduced.

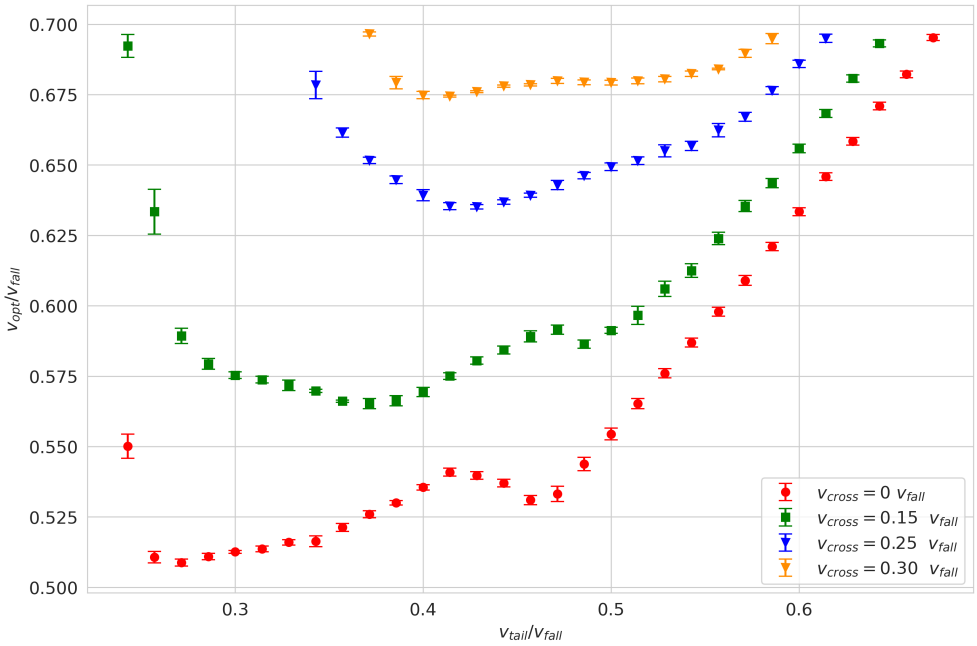

Figure 24: v_{opt} of the walking body as a function of v_{fall} for a few values of v_{cross}.

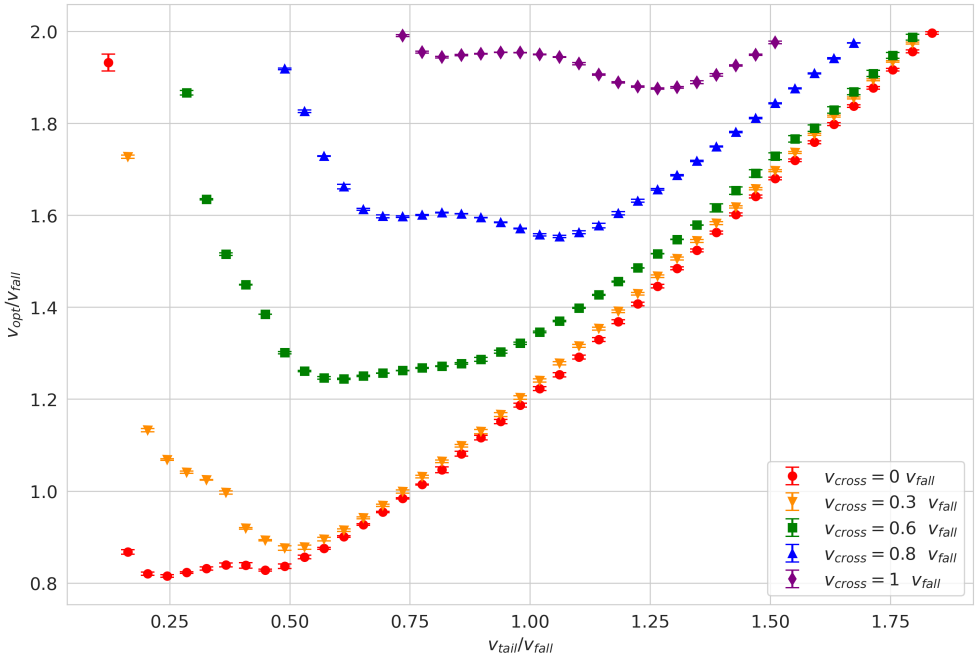

Figure 25: v_{opt} of the running body as a function of v_{fall} for a few values of v_{cross}.

5 Discussion and Conclusion

In the present thesis we study the optimal velocity in which to move under rain falling in a steady wind to get soaked as little as possible. For the first time we address this problem using a model of the human body composed by multiple elementary shapes that move relative to each other, simulating both a walking and a running person. Our model involves spheres, parallelepipeds and capsules. In doing this we also derive a first analytical solution for the wetness of the capsule.

We write a numerical code capable of simulating the movement of our model of the human body during both running and walking, and evaluating the wetness of an arbitrary body made up of spheres, parallelepipeds and capsules, for arbitrary velocity of the wind and the of the body. We check the numerical results obtained this way against the analytical results of the sphere, parallelepiped and capsule, and study the numerical error of our estimations of the wetness for the walking and running bodies, due to the spatial and time discretizations.

We compare our results with the ones obtained by modeling the body as a sphere and as a parallelepiped, and find significant differences: the sphere modeling predicted the optimal velocity to always exist in the presence of a tailwind, and always be larger than twice the falling speed of the rain, but our results show that the optimal speed can be much lower, especially while walking; the parallelepiped modeling predicted the optimal velocity to always equal the tailwind (if present), while our results show that the optimal velocity increases as the crosswind increases.

The general result is that without a tailwind it is better to run as fast as possible (although the advantage of running decreases at larger and larger speed), while in the presence of a strong enough tailwind (at least around a fifth of the falling speed of the rain) there is an optimal walking/running velocity after which accelerating is counterproductive. The presence of a crosswind increases the optimal speed, and may eliminate it completely strong enough. Notably for a limited range of the tailwind and crosswind components it is actually better to walk than to run.

Possible further lines of investigation include a more accurate modeling of the human body and its dynamics, the implementation of different body shapes and their effect on the optimal velocity, and the addition of an umbrella that can protect the body from the rain. The code can be used ”as is” to evaluate the soaking amount for models of arbitrary animals, e.g., dogs or penguins.

References

[1] B. L. Schwartz, M. A. B. Deakin, Math. Mag. **46**, 272 (1973).

[2] D. Hailman, B. Torrents, Math. Mag. **82**, 266 (2009).

[3] F. Bocci, Eur. J. Phys. **33**, 1321 (2012).

[4] T. Kroetz, Rev. Bras. Ensino Fis. **31**, 4304 (2009).

[5] L. Pournin, M. Weber, M. Tsukahara, J. A. Ferrez, M. Ramaioli, T. M. Liebling, Granul. Matter **7**, 119 (2005).

[6] D. C. Lay, S. R. Lay, J. J. McDonald, *Linear algebra and its applications, 5th ed.* (Pearson Education, 2014).

[7] R. Schneider, *Convex bodies: the Brunn–Minkowski theory* (Cambridge Univ. Press, 1993).

[8] P. Huang, S. Pan, Y. Yang, Discrete Comput. Geom. **54**, 728 (2015).

[9] L. da Vinci, *Uomo vitruviano* (1490) (photography by Luc Viatour).

[10] Rasband, W.S., ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA, `https://imagej.net/ij/`, (1997-2018).

[11] A. K. Yegian, Y. Tucker, S. Gillinov, D. E. Lieberman, J. Exp. Biol. **220**, 13 (2019).

[12] T. J. van der Zee , E. M. Mundinger, A. D. Kuo1, Sci Data **9**, 704 (2022).

[13] P. R. Cavanagh, Foot Ankle **7**, 197 (1987).

[14] A. F. dos Santos, T. H. Nakagawa, G. Y. Nakashima, C. D. Maciel, F. Serrão, Int. J. Sports. Med. **37**, 369 (2016).

[15] C. A. Crespi, GitHub repository for the numerical code used in the present thesis, `https://github.com/Cr3sp1/RainSimulation` (2024)

[16] A. S. Nery, N. Nedjah, F. M. G. Franca, Analog. Integr. Circ. Sig. Process. **70**, 189 (2012).

[17] V. Bringi, M. Thurai1, D. Baumgardner, Atmos. Meas. Tech. **11**, 1377 (2018).

[18] V. Billat, A. Demarle, J. Slawinski, M. Paiva, J. Koralsztein, Med. Sci. Sports Exerc. **33**, 2089 (2001).

[19] R. L. Waters, B. R. Lunsford, J. Perry, R. Byrd , J. Orthop. Res. **6**, 215 (1988).