Physics Open 26 (2026) 100381

Contents lists available at ScienceDirect

Physics Open

journal homepage: www.elsevier.com/locate/physo

Optimal speed in rain: A numerical model of a dynamically deforming human body

Cristian Angelo Crespi *, Nicola Manini

Dipartimento di Fisica, Università di Milano, Via Celoria 16, Milano, 20133, Italy

ARTICLE INFO

Handling Editor: Shuang Wu

Dataset link: https://github.com/Cr3sp1/RainSimulation

Keywords:
Human-body modeling
Flux evaluation
Optimum speed

ABSTRACT

The optimal speed for a person running in the rain trying to get soaked as little as possible has long been discussed in the physics and mathematics community. In practice, so far, the human body has mostly been represented as a simple geometric shape such as a parallelepiped or a cylinder. Here we introduce a far more refined model for the complex and dynamically deforming shape of a walking/running person. We introduce a numerical technique akin to ray tracing to evaluate the total amount of intercepted rain with an accuracy never reached in this field before. The main result is that under most wind conditions, running is preferable to walking, with a notable exception occurring when the following conditions a both met: (i) the advancing speed is close to the tailwind and (ii) the crosswind component is negligible.

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 [1–8]. Despite the numerous 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 (following the language adopted in this topic, from here on we will refer to orthogonal parallelepipeds just as "parallelepipeds") 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]. These investigations show that the adopted shape can affect the results dramatically: 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 shape becomes apparent. The main reason this has not been done yet is the difficulty of deriving analytical results for complex shapes. The only exception we are aware of is an analytical solution derived by T. Echekki for a parallelepiped-shaped body covered by a circular umbrella [9].

In the present work we address this problem by means of a numerical approach, based on ray tracing [10,11]. A different numerical approach was attempted by Kroetz [4], but it too only modeled the human body as a parallelepiped, and its focus was rather on the comparison between raindrops placed in a regular array as opposed to random positions. Very recently, Zaegel et al. [12] also implemented a vaguely similar numerical approach, this time modeling the human body with a sphere for the head, and 5 parallelepipeds for the trunk, the arms and the legs. In that work, the mutual positions and angles of these limbs were kept fixed in time: there the focus was mainly the simulation of realistic atmospheric conditions. In the present work we also consider a body model consisting of multiple elementary three-dimensional geometric shapes , but with the limbs modeled in relative motion. We use spheres, parallelepipeds and capsules to model these individual body parts, thus simulating the motion of a walking or running human body far more realistically. A capsule (or spherocylinder) is an elementary three-dimensional shape consisting in a cylinder with hemispherical ends. Capsules have never been studied before in this context: for them we also derive an explicit analytic solution, useful for benchmarking the numerical code.

* Corresponding author.
E-mail address: cristiancrespi1@gmail.com (C.A. Crespi).

https://doi.org/10.1016/j.physo.2026.100381
Received 25 September 2025; Received in revised form 2 February 2026; Accepted 23 February 2026
Available online 4 March 2026
2666-0326/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

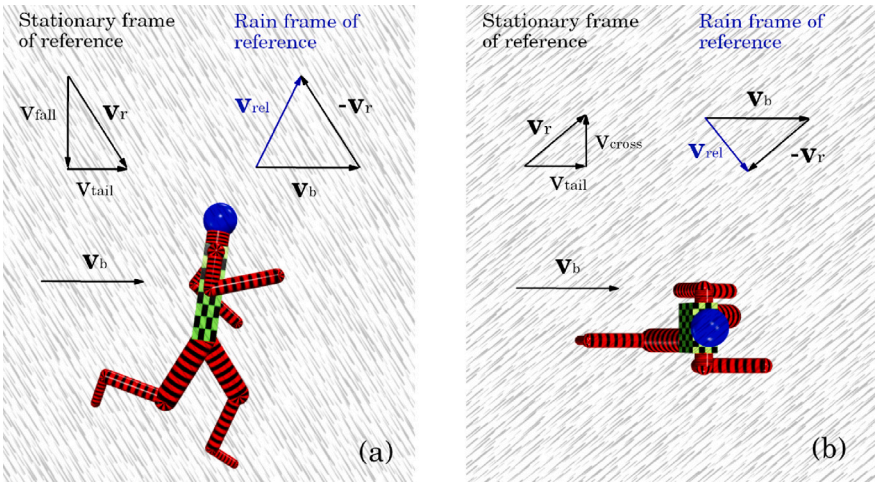

Fig. 1. Model of the human body model viewed from the side (a) and from above (b). The components v_{tail}, v_{cross} and v_{fall} of the velocity $\mathbf{v}_{\text{r}}$ of the raindrops falling at their terminal velocity are indicated in the frame of reference of the stationary ground (black schemes). In the frame of reference of the raindrops falling at their terminal velocity, the body velocity $\mathbf{v}_{\text{b}}$ transforms to the relative-motion velocity $\mathbf{v}_{\text{rel}}$, following Eq. (1) (blue schemes).

2. The model

We model a human body running along a straight path under rainfall in a wind drift. The final target of this modeling is the determination of the optimal speed at which this person should move such that they catch the least possible amount of rain. We assume the following:

1. The ground is horizontal.
2. The path is rectilinear.
3. The raindrops all have the same size and are uniformly distributed in space.
4. The wind velocity is constant in time, and the raindrops have reached their terminal velocity, that includes a horizontal component equal to the wind velocity.
5. The motion of the body consists of a translation at constant speed along the path plus a periodic relative motion that is specific to each body part but shares the same period T.
6. The relative velocity of the body parts due to their periodic motion is negligible compared to the velocity of the rain.
7. The path extension d is so long that any period T of the periodic deformations of the body is small compared to the total time t_{f} taken to traverse the path ($T \ll t_{\text{f}}$).
8. All involved speeds are negligible compared to the speed of light, so that the nonrelativistic limit and Galilean transformations apply throughout.

We adopt a rest frame of reference with the x axis aligned with the horizontal advancement direction of the body, and with the y and z axes in the perpendicular horizontal and vertical directions, respectively. According to assumptions 1, 2 and 5, the body is translating with a velocity $\mathbf{v}_{\text{b}} = v_{\text{b}}\hat{\mathbf{e}}_x$, and the rain has a velocity $\mathbf{v}_{\text{r}}$. The z component of $\mathbf{v}_{\text{r}}$ is negative, and we refer to its absolute value as the falling velocity v_{fall}. We refer to the lateral (y) component of $\mathbf{v}_{\text{r}}$ as the "crosswind" v_{cross}. Since all the static bodies we take into consideration are symmetric under the transformation $y \to -y$, and so are the dynamic ones once we take the time average, we consider only positive values of v_{cross}. We refer to the value of the x component of $\mathbf{v}_{\text{r}}$ as the "tailwind" v_{tail}. We indicate the total length of the path traversed with $d = v_{\text{b}}\, t_{\text{f}}$.

The volume of water W hitting the advancing body over the entire walk equals the number of raindrop hits times the volume of each drop. Thanks to our assumption 3 we can define a dimensionless "rain density" $\rho_{\text{rain}} = N_{\text{drop}} V_{\text{drop}}/V$, where N_{drop} is the number of raindrops contained in a macroscopic volume V, and V_{drop} is the volume of a single drop. In short ρ_{rain} is the volume fraction of liquid water freely falling in the atmosphere.

Consider now the reference frame in which the raindrops are stationary. In this reference frame, the velocity of the body is:

$$\mathbf{v}_{\text{rel}} = \mathbf{v}_{\text{b}} - \mathbf{v}_{\text{r}}\,, \tag{1}$$

as shown in Fig. 1. The body advances with velocity $\mathbf{v}_{\text{rel}}$ through the stationary raindrops sweeping up a volume V_{sweep} in its motion. The collected water amount is then

$$W = \rho_{\text{rain}}\, V_{\text{sweep}}\,. \tag{2}$$

Since the volume fraction ρ_{rain} is a fixed constant, our problem reduces to evaluating (and minimizing) V_{sweep}. As derived in Ref. [2], for a body executing a rigid translation at velocity $\mathbf{v}_{\text{rel}}$, the total swept volume V_{sweep} over a path of length d is given by:

$$V_{\text{sweep}}(\mathbf{v}_{\text{rel}}) = S_{\text{b}}(\mathbf{v}_{\text{rel}})\, \frac{\|\mathbf{v}_{\text{rel}}\|}{v_{\text{b}}}\, d\,, \tag{3}$$

where $S_{\text{b}}(\mathbf{v}_{\text{rel}})$ is the surface area of the projection of the body on a plane perpendicular to $\mathbf{v}_{\text{rel}}$, as shown in Fig. 2.

The only nontrivial part of the problem is then the determination of the dependence of S_{b} on $\mathbf{v}_{\text{rel}}$. Note that (except for few very specially oriented low-dimensional shapes) both limits $\lim_{v_{\text{b}}\to 0^+} V_{\text{sweep}}(\mathbf{v}_{\text{rel}})$ and $\lim_{v_{\text{b}}\to+\infty} V_{\text{sweep}}(\mathbf{v}_{\text{rel}})$ are nonzero for any $\mathbf{v}_{\text{r}}$. In particular, due to the finiteness of the $v_{\text{b}} \to 0^+$ limit of both $\mathbf{v}_{rel}$ and $S_{\text{b}}(\mathbf{v}_{rel})$, $V_{\text{sweep}}(\mathbf{v}_{\text{rel}})$ diverges as $v_{\text{b}} \to 0^+$ because of the v_{b} term at the denominator of Eq. (3): as a result, ordinary three-dimensional body shapes lead to unlimited growth of the accumulated water W as the advancement speed decreases to zero. In the opposite large-v_{b} limit, $\mathbf{v}_{\text{rel}}$ approaches $v_{\text{b}}\hat{\mathbf{e}}_x$, and therefore Eq. (3) simplifies to

$$\lim_{v_{\text{b}}\to+\infty} V_{\text{sweep}}(\mathbf{v}_{\text{rel}}) = S_{\text{b}}(\hat{\mathbf{e}}_x)\; d\,. \tag{4}$$

This constant large-speed limit (independent of the rain velocity $\mathbf{v}_{\text{r}}$) can sometimes turn out as the optimal condition, typically in wind conditions that make $V_{\text{sweep}}(\mathbf{v}_{\text{rel}})$ a monotonically decreasing function of v_{b}.

We proceed to extend these expressions to a dynamical body, i.e. a body that changes shape as a function of time. Since the velocity of each body part relative to the body center of mass is small compared to the velocity of the rain (assumption 6) we can approximate the instantaneous velocity of each body part with the velocity of the whole body $\mathbf{v}_{\text{b}}$, i.e. $\mathbf{v}_{\text{rel}}$ in the rain rest frame. This way we can write the area of the projection of the whole body as a function uniquely on the velocity

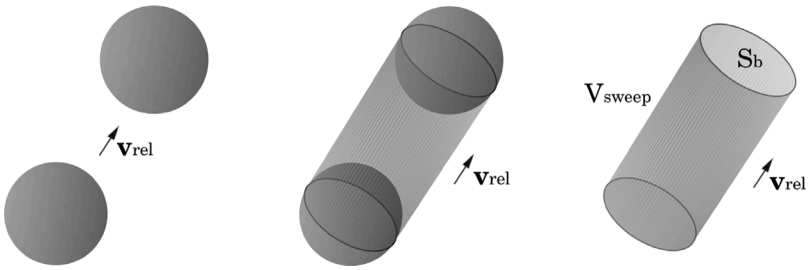

Fig. 2. The volume V_{sweep} swept by a sphere advancing with velocity $\mathbf{v}_{\text{rel}}$ across stationary rain. The sphere's orthogonal projection on the plane perpendicular to $\mathbf{v}_{\text{rel}}$ is a disk with area S_b and V_{sweep} is the volume of a cylinder with surface area S_b and height d.

of the entire body and of the instantaneous location and orientation of each body part. For a given body, the latter is a function of time, and so is the projected area $\tilde{S}_b(\mathbf{v}_{\text{rel}}, t)$. We can then generalize Eq. (3) by replacing the advancement time $t_f = d/v_b$ with a time integration over the $[0, t_f]$ interval:

$$V_{\text{sweep}}(\mathbf{v}_{\text{rel}}) = \|\mathbf{v}_{\text{rel}}\| \int_0^{t_f} dt\, \tilde{S}_b(\mathbf{v}_{\text{rel}}, t)\,. \tag{5}$$

Assuming that the time spent in the path is an integer multiple of the period T of the body deformation (reflected in the periodic function $\tilde{S}_b(\mathbf{v}_{\text{rel}}, t)$), we can replace $\tilde{S}_b(\mathbf{v}_{\text{rel}}, t)$ with its time average over one period

$$S_b(\mathbf{v}_{\text{rel}}) = \frac{1}{T}\int_0^T dt\, \tilde{S}_b(\mathbf{v}_{\text{rel}}, t)\,. \tag{6}$$

We then use this time average to extend Eq. (3) to dynamical bodies, provided that assumptions 5, 6, and 7 are satisfied.

Since we cannot solve the dynamic body analytically we approximate $S_b(\mathbf{v}_{\text{rel}})$ with its discrete time average:

$$S_b(\mathbf{v}_{\text{rel}}) \simeq \frac{1}{N}\sum_{i=0}^{N-1} \tilde{S}_b\left(\mathbf{v}_{\text{rel}},\ i\frac{T}{N}\right). \tag{7}$$

Furthermore, since both d and ρ_{rain} amount to fixed and essentially arbitrary prefactors, in the following we omit them, and evaluate and minimize the ratio of the swept volume $V_{\text{sweep}}(\mathbf{v}_{\text{rel}})$ to the path length d. This ratio represents the relative wetness:

$$R_b(\mathbf{v}_{\text{rel}}) = \frac{V_{\text{sweep}}(\mathbf{v}_{\text{rel}})}{d} = S_b(\mathbf{v}_{\text{rel}})\,\frac{\|\mathbf{v}_{\text{rel}}\|}{v_b}\,. \tag{8}$$

For given $\mathbf{v}_r$, R_b can be rewritten as a function of v_b using Eq. (1) and $\mathbf{v}_b = v_b\hat{\mathbf{e}}_x$:

$$R_b(v_b) = R_b(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)\,. \tag{9}$$

Due to Eq. (4) it is clear that

$$\lim_{v_b\to+\infty} R_b(v_b) = S_b(\hat{\mathbf{e}}_x)\,. \tag{10}$$

Minimizing W is equivalent to minimizing $R_b(v_b)$.

For the sake of convenience, we will express all velocities in units of v_{fall}. For R_b we adopt SI units (m^2).

2.1. Analytic results

We build our model for the human body in terms of a few elementary geometric shapes, namely spheres, parallelepipeds and capsules. The evaluation of R_b relies on the projection of these elementary shapes onto arbitrary planes. Analytic projection formulas turn out useful as benchmarks for the numerical evaluation.

First we recall the formulas for the orthogonal projection onto planes perpendicular to a given vector $\mathbf{v}$. For definiteness we consider planes passing through the origin. The projector operator $P_{\mathbf{v}}$ onto a plane through the origin and perpendicular to $\mathbf{v}$ projects a point, represented by vector $\mathbf{b}$, as follows [13]:

$$P_{\mathbf{v}}(\mathbf{b}) = \mathbf{b} - \frac{\mathbf{b}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v}. \tag{11}$$

For a body B defined as a set of points in $\mathbb{R}^3$, its projection is $P_{\mathbf{v}}(B) = \{P_{\mathbf{v}}(\mathbf{b}) : \mathbf{b}\in B\}$.

2.1.1. The sphere and the parallelepiped

We now summarize a few well established results for spherical and parallelepiped shapes.

The orthogonal projection of a sphere of radius r and center $\mathbf{c}$ on an arbitrary plane is a disk with radius r and center $P_{\mathbf{v}}(\mathbf{c})$ [2,3]. This observation leads to the following formula for the swept-volume to path-length ratio $R_b(v_b)$:

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

For a sphere the condition for a finite v_{opt} to exist is simply $v_{\text{tail}} > 0$. For positive tailwind the value of this optimal velocity is:

$$v_{\text{opt}} = \frac{\|\mathbf{v}_r\|^2}{v_{\text{tail}}}\,. \tag{13}$$

As discussed in Refs. [1,3], when dealing with a parallelepiped one only needs to consider at most three faces, depending on orientations of the parallelepiped relative to $\mathbf{v}$. We identify these faces with the vectors $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3$: each vector is perpendicular to the face it represents and its magnitude is equal the area of that face. The orthogonal projection of the whole parallelepiped equals the sum of the projections of these faces. The projection of a face is a parallelogram whose vertices are the projections of the vertices of the face. Since these faces are adjacent, so are their projections. As a result, the projection of the parallelepiped is generally the hexagon composed of these three adjacent parallelograms. This leads to the following formula for $R_b(v_b)$:

$$R_b(v_b) = \left[\sum_{i=1}^{3}\|\mathbf{S}_i\cdot(v_b\hat{\mathbf{e}}_x - \mathbf{v}_r)\|\right]\frac{1}{v_b}. \tag{14}$$

For rectangular parallelepipeds with sides parallel to the axes, Eq. (14) is straightforwardly expressed in terms of the areas S_x, S_y and S_z of the faces perpendicular to the x, y and z axis respectively:

$$R_b(v_b) = \frac{S_x\,|v_{\text{tail}} - v_b| + S_y\,|v_{\text{cross}}| + S_z\,v_{\text{fall}}}{v_b}. \tag{15}$$

Then, the condition for a finite v_{opt} is the following [1,3]:

$$v_{\text{tail}} > \frac{S_y\,|v_{\text{cross}}| + S_z\,v_{\text{fall}}}{S_x}. \tag{16}$$

When this condition is satisfied, $v_{\text{opt}} \equiv v_{\text{tail}}$.

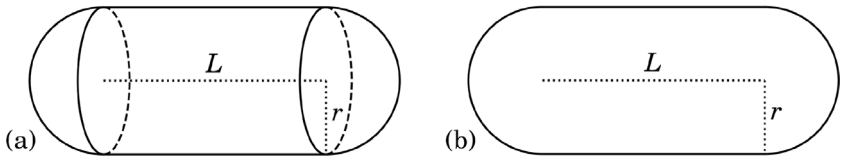

Fig. 3. (a) Perspective view of a capsule with axis L and radius r. (b) A 2D stadium, with axis L and radius r.

2.1.2. The capsule

In the context of human-body representation, the capsule, illustrated in Fig. 3a, is certainly a relevant shape. We derive the analytic expression for the wetness of a capsule. A capsule is defined as follows [14]: let L be a line segment in $\mathbb{R}^3$ and r a positive length. The capsule with axis L and radius r is the set of points whose distance from L is smaller than 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. We recall that, given two sets of points A and B in Euclidean space, their Minkowski sum is the set $A + B := \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\}$ [15].

For the evaluation of the orthogonal projection $P_{\mathbf{v}}$ of a capsule C on a plane perpendicular to a given vector $\mathbf{v}$, we use the fact that the projection operation distributes over the Minkowski sum, i.e. $P_{\mathbf{v}}(A + B) = P_{\mathbf{v}}(A) + P_{\mathbf{v}}(B)$. Indeed, using Eq. (11) for individual vectors/points we verify that

$$P_{\mathbf{v}}(\mathbf{a}+\mathbf{b}) = \mathbf{a}+\mathbf{b}-\frac{(\mathbf{a}+\mathbf{b})\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v} = \mathbf{a}-\frac{\mathbf{a}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v}+\mathbf{b}-\frac{\mathbf{b}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v} = P_{\mathbf{v}}(\mathbf{a})+P_{\mathbf{v}}(\mathbf{b})\,. \quad (17)$$

Therefore, for two sets,

$$P_{\mathbf{v}}(A+B) = \{P_{\mathbf{v}}(\mathbf{a}+\mathbf{b}) : \mathbf{a}\in A,\, \mathbf{b}\in B\} = \{P_{\mathbf{v}}(\mathbf{a}) + P_{\mathbf{v}}(\mathbf{b}) : \mathbf{a}\in A,\, \mathbf{b}\in B\} = \quad (18)$$

$$= \{\hat{\mathbf{a}}+\hat{\mathbf{b}} : \hat{\mathbf{a}}\in P_{\mathbf{v}}(A),\, \hat{\mathbf{b}}\in P_{\mathbf{v}}(B)\} = P_{\mathbf{v}}(A) + P_{\mathbf{v}}(B)\,.$$

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 $P_{\mathbf{v}}(\mathbf{l}_2)$. 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. Therefore, the projection of a capsule $C = L + S$ is $\hat{C} = \hat{L} + \hat{S}$. The Minkowski sum between a line segment and a disk is a stadium [16], also called "sausage body", namely the two-dimensional version of a capsule: a stadium with axis L and radius r is the set of points whose distance from L is smaller or equal to r, see Fig. 3b. The area of a stadium can be easily derived observing that it is made up of two half disks with radius r plus a rectangle with sides $2r$ and l, where $l = |\mathbf{l}_2 - \mathbf{l}_1|$ is the length of L:

$$S_s = \pi r^2 + 2rl. \quad (19)$$

We apply Eq. (19) to the stadium obtained from projecting a capsule: r is the same as that of the capsule, and the stadium length $\hat{l}$ is obtained by projecting the vector $\Delta\mathbf{l} := \mathbf{l}_2 - \mathbf{l}_1$.

$$\hat{l} = \|P_{\mathbf{v}}(\mathbf{l}_2) - P_{\mathbf{v}}(\mathbf{l}_1)\| = \|P_{\mathbf{v}}(\Delta\mathbf{l})\| = \left\|\Delta\mathbf{l} - \frac{\Delta\mathbf{l}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v}\right\|. \quad (20)$$

With this ingredient we calculate the projection S_c of a capsule on a plane perpendicular to its velocity $\mathbf{v}_{\rm rel}$ relative to the rain. Based on Eqs. (19) and (20) we obtain:

$$S_c(\mathbf{v}_{\rm rel}) = \pi r^2 + 2r\left\|\Delta\mathbf{l} - \frac{\Delta\mathbf{l}\cdot\mathbf{v}_{\rm rel}}{\mathbf{v}_{\rm rel}\cdot\mathbf{v}_{\rm rel}}\mathbf{v}_{\rm rel}\right\|. \quad (21)$$

Substitution in Eq. (8) provides the volume/length ratio $R_{\rm b}$ for a capsule:

$$R_{\rm b}(\mathbf{v}_{\rm rel}) = \left[\pi r^2 + 2r\left\|\Delta\mathbf{l} - \frac{\Delta\mathbf{l}\cdot\mathbf{v}_{\rm rel}}{\mathbf{v}_{\rm rel}\cdot\mathbf{v}_{\rm rel}}\mathbf{v}_{\rm rel}\right\|\right]\frac{\|\mathbf{v}_{\rm rel}\|}{v_{\rm b}}\,. \quad (22)$$

2.2. The full human-body model

Considering the significant flexibility of the human body and the variability in shape and size ratios (e.g. the body mass index [17]) exhibited by human beings, constructing a fully general model seems an impossible task. In this work we propose a relatively simple, but easy to refine model for the human body, sketched in Fig. 1. Rather than following statistical data [18], we construct a model based on the Vitruvian Man by L. da Vinci [19], see Fig. 4. By analyzing the text accompanying the drawing and measuring the drawing itself, for a body with total height h we obtain the following estimations:

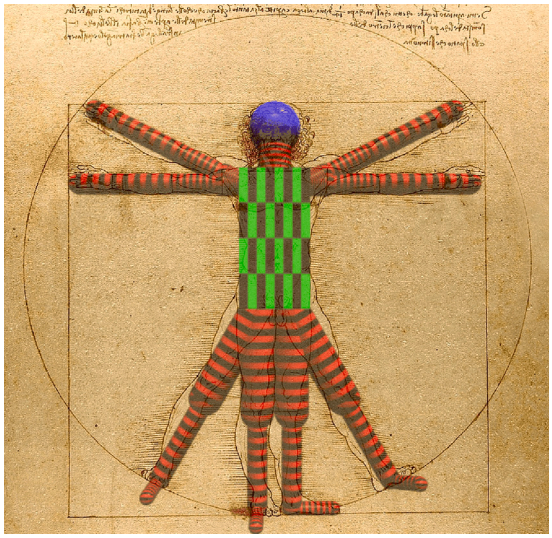

Fig. 4. Our model of the human body overlayed on the Vitruvian Man by Leonardo da Vinci (photography by Luc Viatour). The colors and patterns denote the shape of the base body part: solid blue for sphere, checkered green/black for parallelepiped, and red/black stripes for capsule.

- The head is $h/8$ tall.
- The neck is $h/24$ long and $h/14$ thick.
- The torso is $h/3$ tall, $h/6$ wide and $h/12$ thick.
- Each shoulder is $h/24$ long and $0.06\,h$ thick.
- Each upper arm is $h/8$ long and $h/20$ thick.
- Each forearm, including its hand, is $h/4$ long and $h/20$ thick.
- Each thigh is $h/4$ long and $h/12$ thick.
- Calves are $h/4$ long and $h/20$ thick.
- Each foot is $h/7$ long and $h/30$ thick.

We adopt $h = 1.68$ m as a plausible height for a person. The swept-volume to path-length ratio is clearly proportional to h^2. However, our target, namely the optimal speed, is independent of the overall scale of the body (namely of h), but is purely a function of its shape.

As anticipated, we build our body using as base shapes spheres, parallelepipeds and capsules. For the torso we use a rectangular parallelepiped with sides $s_x = h/12$, $s_y = h/6$, $s_z = h/3$. We represent the neck with a capsule of radius $r = 1/24\,h$ and length $\|\Delta\mathbf{l}\| = 5/48\,h$. The head is modeled as a sphere of radius $r = h/16$. The neck connects the torso to the head: one endpoint $\mathbf{l}_1$ is placed at 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 rigidly fixed relative to the

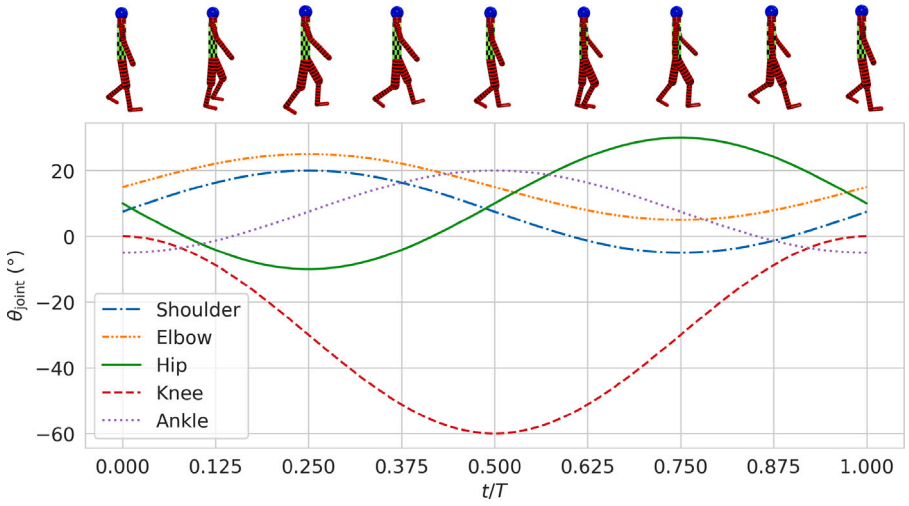

Fig. 5. Time evolution of the angles θ_{joint} of the model parts representing individual limbs at the right side of the body during walking. For the limbs at the left-side, the phases are shifted by 180°. Upper images: illustrative snapshots of the resulting human walking model at intervals $T/8$.

torso. All other body elements (shoulders, upper arms, forearms, thighs, calves, and feet) are modeled by capsules with specified radii and lengths. The shoulders have length $\|\Delta\mathbf{l}\| = h/24$ and radius $r = 0.03\,h$. Each of them has one of its endpoints situated on one of the sides of the torso, $0.03\,h$ below the middle of the top side of that face; their axis is perpendicular to that face. The second endpoint of each shoulder coincides with the first endpoint of one of the two capsules of length $\|\Delta\mathbf{l}\| = h/8$ and radius $r = h/40$ 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 $r = h/40$ that represent the forearms. The shoulders are rigidly fixed to the torso during both walking and running. The upper arms can rotate around the endpoint connecting them to the shoulders. This movement induces a roto-translation of the forearms, which can further rotate around the endpoint they share with the upper arms. The capsules modeling the thighs have length $\|\Delta\mathbf{l}\| = h/4$ and radius $r = h/24$. They both have their first endpoint 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 $r = h/40$, that represent the calves. The feet have length $\|\Delta\mathbf{l}\| = 23/210\,h$ and radius $r = h/60$. Feet capsules are connected to the calves at one endpoint, positioned $r_{\text{calf}} - r_{\text{foot}} = h/120$ below the endpoint of each calf, so to prevent the calf from protruding below the foot. The thighs rotate around the endpoint connecting them to the torso, taking the calves and feet with them. The calves rotate together around the endpoint connecting them to the thighs, moving the attached foot. Each foot rotate around the endpoint connecting it to the respective calf. Fig. 4 displays a direct comparison between our model of the body and the Vitruvian Man by Da Vinci. Full numerical details of our body are available on GitHub [20].

This model, although idealized, provides a fair approximation of the human overall shape. If a more refined description was desirable, it is straightforward to add extra elements among those available (e.g. capsules) to the model. For example, the tapering of a leg could be modeled by combining a few coaxial capsules of decreasing radius and increasing length. Deviations from a circular section can be achieved by combining non-coaxial capsules.

We come to describe how the model changes shape in time. We focus on two regimes of motion: walking and running. We model both by means of periodic roto-translations of the limbs around their respective joints. We write the periodic time evolution of the angle θ_{joint} of each joint as a Fourier series:

$$\theta_{\text{joint}}(t) = \theta_0 + \sum_{n=1}^{N_{\max}} \theta_{\text{a}\,n} \sin\left(2\pi n \frac{t}{T} + \phi_n\right) . \tag{23}$$

Table 1
Joint-articulation angular parameters for walking and running, see Eq. (23). The reported time-offset phases ϕ apply to the joints at the right side of the body: those of the left side are incremented by π.

Motion	Walking		Running		Both
Joint	θ_0	$\theta_{\text{a}\,1}$	θ_0	$\theta_{\text{a}\,1}$	ϕ
shoulder	7.5°	12.5°	−15°	15°	0°
elbow	15°	10°	97.5°	12.5°	0°
hip	10°	20°	7.5°	32.5°	180°
knee	−30°	30°	−67.5°	52.5°	90°
ankle	7.5°	12.5°	−2.5°	22.5°	−90°

In this work we limit ourself to only the first term non-constant term, thus we only consider $N_{\max} = 1$, but it would be straightforward to add additional terms if required for a more accurate description of motion. For realistic amplitudes and relative phases of these sinusoids, we refer to articles on the biomechanics of walking and running [21–23]. For each joint, we conventionally consider the angle $\theta_{\text{joint}} = 0$ when the body is standing with straight legs and the arms parallel to the torso and when the feet are perpendicular to the legs. We further assume that $\theta_{\text{joint}} > 0$ when a limb rotates advancing in the direction of motion of the body, and in the case of the ankle when the foot rotates upward toward the shin. Note that the time-offset phases ϕ coincide for running and walking. The adopted end angles and time-offset phases are reported in Table 1 for all joints. During walking, the torso stands vertical, i.e. parallel to the z axis, while during running it forms a constant forward angle of 8° [23].

Fig. 5 illustrates the time evolution of the angles θ_{joint} over a period T for walking. Fig. 6 reports the same quantities for running. Both figures include a sequence of snapshots of the human model over one period. Full numerical details of both walking and running bodies are provided in GitHub [20].

2.3. Numerical details

We have all tools needed to determine R_{b} as a function of v_{b}, and thus v_{opt}, for both the walking and the running body, under varied conditions of v_{tail} and v_{cross}. We first identify a range of speeds $0 \le v_{\text{b}} \le v_{\max}$ that we deem acceptable for running and walking. Since we measure velocities in units of v_{fall} we first need to assess its actual order of magnitude. Rain falls at different speeds depending on the size of the raindrops, with a minimum speed for precipitations around 2.5 m/s [24]. As an upper bound for the running speed we consider the

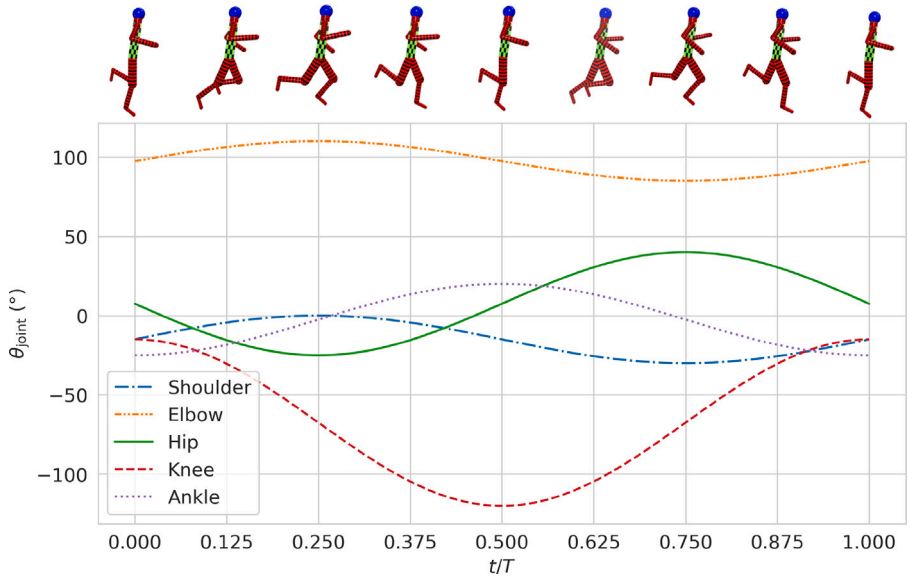

Fig. 6. Same as Fig. 5 but for running instead of walking.

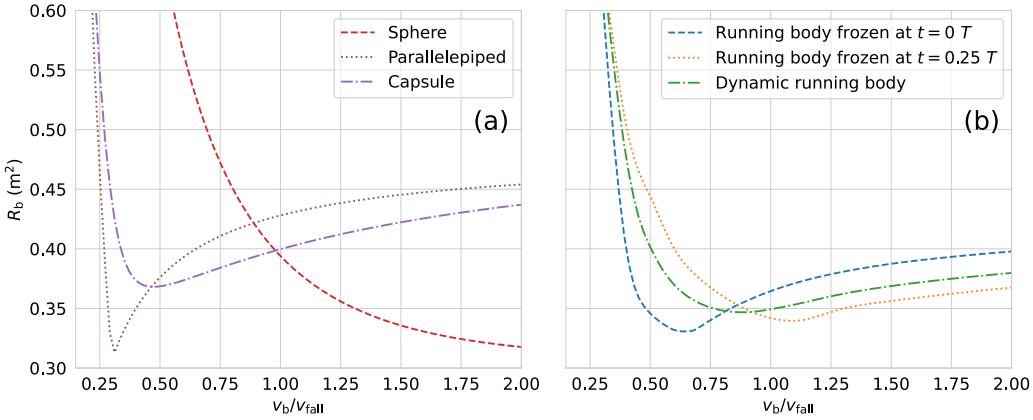

Fig. 7. Comparison among the wetness amount R_b, as a function of the body advancement speed v_b, (a) for a sphere, a parallelepiped, and a capsule; and (b) for the running body frozen at two configurations (those at times $t = 0$ and $0.25\,T$, see snapshots in Fig. 6) and for the fully dynamic running body. The sphere has radius $r = 0.32$ m, the parallelepiped has sides $s_x = 0.2$ m, $s_y = 0.3$ m, $s_z = 1.6$ m, and the capsule has radius $r = 0.17$ m and axis $\Delta\mathbf{l} = (0, 0, 1.2)$ m. The wind components are $v_{tail} = 0.3\,v_{fall}$ and $v_{cross} = 0.1\,v_{fall}$.

average speed of an elite marathon runner, namely around 5 m/s [25], corresponding to at most $v_{max} = 2\,v_{fall}$. Furthermore, a top speed for a healthy walking adult is approximately 1.75 m/s [26], namely no more than $v_{max} = 0.7\,v_{fall}$.

As detailed in Appendix A, we construct a simulation code to evaluate $R_b(v_b)$ as a function of the body speed v_b, in the appropriate range $0 \le v_b \le v_{max}$. The code averages $R_b(v_b)$ over the motion of the joints, Eq. (23), making N discrete time steps spaced by $dt = T/N$. For the space and time discretization parameters, we adopt $dx = 0.001$ m and $dt = 0.02\,T$, after checking that the results change very little for smaller values, see Appendices B and C.

Fig. 7 compares the wetting amount R_b as a function of the body speed for different body shapes. Simple geometric shapes are reported in panel (a); panel (b) illustrates the effect of the body shape dynamical evolution on R_b for the running body. While the overall dependence and values of R_b as a function of v_b are fairly similar for the dynamically evolving body, for the frozen body, and for most simple geometric objects, we see a significant spread in the speed v_{opt} at which R_b is minimum. Indeed v_{opt} varies between $\simeq 0.6\ v_{fall}$ and $\simeq 1.1\ v_{fall}$, indicating that a realistic dynamic-body simulation is needed for an accurate estimation of v_{opt}.

3. Results

For given v_{tail} and v_{cross}, we initially apply a standard Brent minimization to the numerical function $R_b(v_b)$[27] to identify a first estimate to its minimum. Unfortunately, $R_b(v_b)$ is affected by small but not entirely negligible numerical noise, as illustrated in Fig. 8. One of the noise-associated wiggles can trap the Brent minimization, leading to a poor estimation of v_{opt}. To refine the Brent estimate, we further fit a parabola on a few values of $R_b(v_b)$ around the initial Brent minimum:

$$P(v_b) = R_{min} + k(v_b - v_{opt})^2. \tag{24}$$

In concrete, we evaluate $R_b(v_b)$ at N_{fit} points spaced by dv around this initial estimate of v_{opt} and fit these data points with Eq. (24) by means of a least-squares regression. As illustrated by Fig. 8, the parabola averages over the numeric noise: as a result, its minimum provides a more reliable estimate of v_{opt} and R_{min}, along with estimates of their uncertainties. In conditions where the fit locates v_{opt} outside the range $(0, v_{max}]$, or fails to locate a minimum (i.e. $k \le 0$), we assign $v_{opt} = v_{max}$ and $R_{min} = R_b(v_{max})$. Further details on this minimization method are discussed in Appendix A.4. For all evaluations of v_{opt} and R_{min} discussed in this Section, we adopt $N_{fit} = 9$ and $dv = 0.006\,v_{fall}$

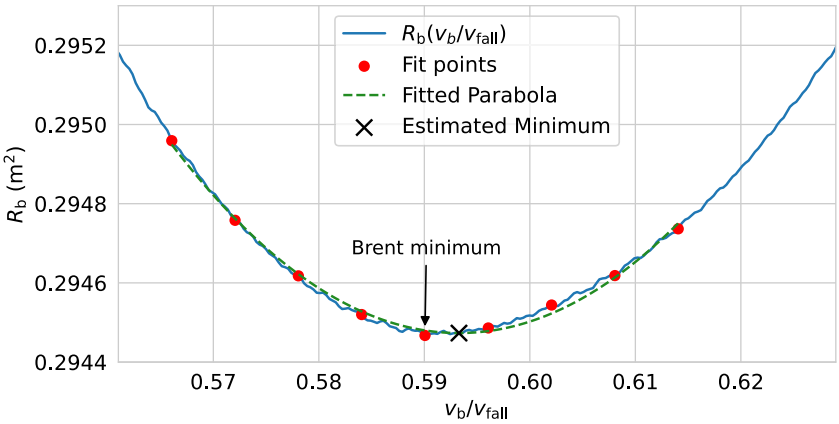

Fig. 8. Example of the evaluation of v_{opt} for a walking body, with the following conditions: $v_{tail} = 0.5\ v_{fall}$, $v_{cross} = 0.15\ v_{fall}$, $dx = 0.001$ m, $N = 50$, $N_{fit} = 9$, $dv = 0.006\, v_{fall}$. Solid blue line: the relative wetness $R_b(v_b)$ evaluated numerically over a very fine mesh of velocities, illustrating how this quantity is affected by numerical noise. Red dots: the few evaluated points $R_b(v_b)$ that we use to improve the minimum estimation through a parabolic fit, namely the dashed green line. Black cross: the minimum of $R_b(v_b)$ estimated by the fitted parabola, and providing an estimation of v_{opt} less affected by the numerical noise.

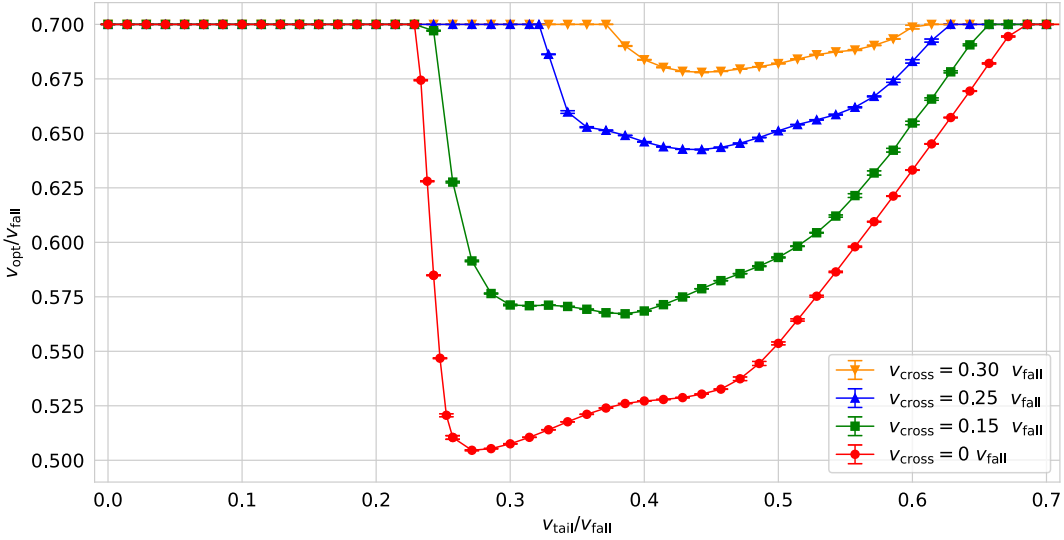

Fig. 9. The best advancing speed v_{opt} of the walking-body model as a function of the tailwind speed v_{tail}, for a few values of v_{cross}. Each point is obtained through a minimization of the kind illustrated in Fig. 8 and described in Appendix A.4.

Figs. 9 and 10 report the resulting dependence of v_{opt} on the tailwind speed v_{tail}, for a few values of v_{cross}. We see that for both the walking-body model (Fig. 9) and the running-body model (Fig. 10), the effect the cross wind is to systematically increase v_{opt}, and to shrink the range of v_{tail} where the minimum is located at a speed different from v_{max}. A sufficiently large v_{cross} can even eliminate the minimum altogether, although this effect becomes less dramatic for larger v_{tail}. This crosswind dependence is fully expected: for very large crosswind most rain is intercepted from the body side, so that advancing faster leads to a shorter exposure, and eventually to less soaking. Furthermore, the comparison between Figs. 9 and 10 indicates that the walking-body model achieves systematically smaller values $0.5\, v_{fall} \le v_{opt} \le 0.7\, v_{fall}$ than the values $0.8\, v_{fall} \le v_{opt} \le 2\, v_{fall}$ predicted by the running-body model. For small v_{tail}, Figs. 9 and 10 exhibit sharp increases of v_{opt} as v_{tail} is reduced toward its smallest relevant values.

It is useful to compare these results with the analytic optimal speed for a sphere and a straight parallelepiped. Fig. 11 reports this comparison for two values of v_{cross}. The optimal speed of the walking and running human-body models exhibits major qualitative and quantitative differences with both the sphere and the parallelepiped. Under no wind condition the sphere's v_{opt} happens to be smaller than $2\, v_{fall}$, indicating that it systematically exceeds v_{max} for both the walking and running body. If a sphere was an accurate model of a human body, then running at the fastest practical speed would always be the best option to minimize soaking. In contrast, the straight parallelepiped model predicts that, when $R_b(v_b)$ has a minimum, it occurs at $v_b = v_{tail}$. We see that our detailed human-body model contrasts both these conclusions: v_{opt} takes nontrivial values well below $2\, v_{fall}$, and generally larger than v_{tail}, across a broad range of v_{tail} and v_{cross}. Interestingly, when there exists a minimum inside the considered velocity range, the optimum speed of both the walking-body and running-body models always ends up in between the predictions of the parallelepiped and the sphere models.

One common property shared by all the models considered is the necessity of having $v_{tail} > 0$ to actually observe a minimum of $R_b(v_b)$. Although a priori nothing prevents the walking-body and running-body models from predicting a minimum of $R_b(v_b)$ even for negative v_{tail}, in practice we found none in either model, at least across the realistic $[0, v_{max}]$ range. This result indicates that in case of headwind, the fastest run is always the best strategy.

It is useful to analyze how v_{opt} depends on both horizontal wind components v_{tail} and v_{cross} on the same footing. To this purpose,

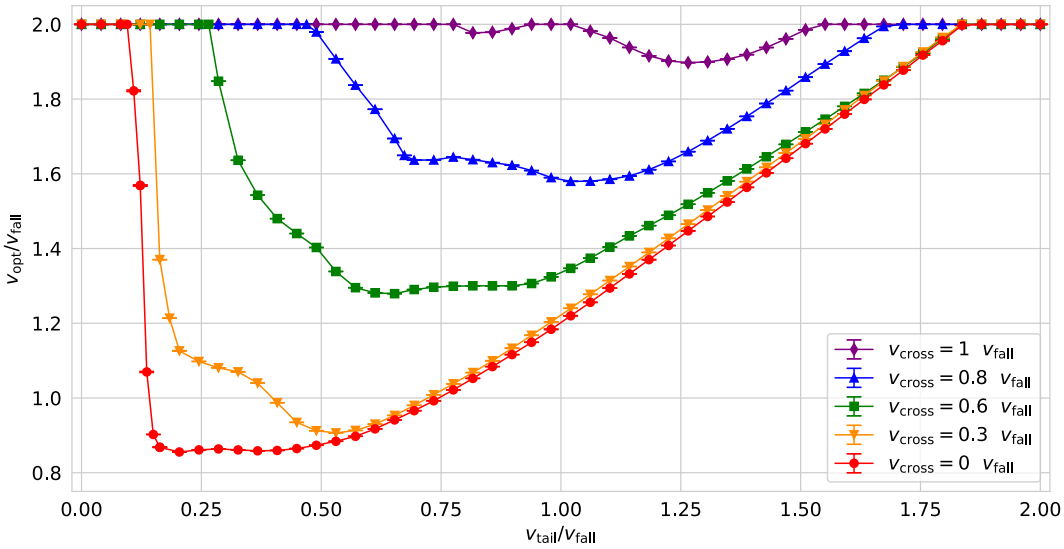

Fig. 10. Same as Fig. 9, but for the running-body model, with $v_{max} = 2\, v_{fall}$.

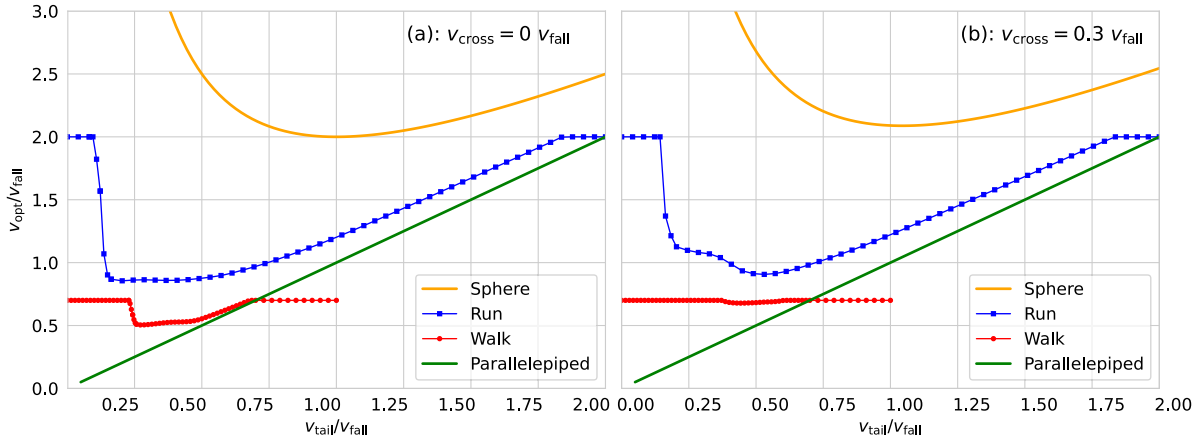

Fig. 11. Comparison of the best advancing speed v_{opt} for our models of walking (circles) and running (squares) human body, with those of a sphere and of a straight parallelepiped, as a function of the tailwind speed v_{tail}, for two values of the crosswind speed: (a) $v_{cross} = 0$; (b) $v_{cross} = 0.3\, v_{fall}$. Numerical error bars on the data are smaller than the dot size.

for both models Fig. 12 reports v_{opt} as 2D colormaps. This figure shows that, regardless of the value of v_{tail}, for both models v_{opt} is a monotonically increasing function of v_{cross}. This confirms the previous observation that v_{cross} increases v_{opt}, to the extent that a large enough v_{cross} can even suppress the minimum altogether.

The computational method outlined in this work puts us in a position to address the question of whether walking or running is preferable when it rains. Common sense may suggest that running is always preferable, but our model allows us to check this assumption quantitatively. Fig. 13 compares the intercepted rain $R_b(v_b)$ for the walking and running body, and for different wind conditions reported in different panels. It is apparent that walking can lead to a better performance than running for suitable conditions, as illustrated by panel (a). To make this investigation more systematic, we compare the values of the optimum wetness $R_{walk} \equiv R_b(v_{opt})$ of the walking-body model and the analogous quantity R_{run} for the running-body model, at given wind components v_{tail} and v_{cross}, and check which one is smaller. To describe the relative advantage of one advancement pattern compared to the other, we define

$$\Delta R_b\% = 100 \times \frac{R_{walk} - R_{run}}{R_{run}}\,. \tag{25}$$

Table 2
Average projected surfaces of the projections of the walking and running models on planes perpendicular to the Cartesian axes x, y and z.

Motion	$S_b(\hat{\mathbf{e}}_x)$ (m^2)	$S_b(\hat{\mathbf{e}}_y)$ (m^2)	$S_b(\hat{\mathbf{e}}_z)$ (m^2)
walking	0.493	0.297	0.140
running	0.429	0.359	0.200

This quantity represents, in percent, how much extra wet a person becomes when walking instead of running. ΔR_b is usually positive, but whenever it turns negative, it becomes advantageous to walk rather than run. Fig. 14 reports the relevant comparison. In the vast majority of horizontal wind velocity components the best course of action is running, either as fast as possible or at a nontrivial speed v_{opt}. However, Fig. 14 identifies a small but relevant region of wind velocities where walking is preferable (negative ΔR_b), highlighted by circular dots and exemplified by Fig. 13(a). The speeds are $v_{tail} \approx 0.7\, v_{fall}$ and $v_{cross} < 0.2\, v_{fall}$. For specific values of v_{tail} and v_{fall}, walking can result in collecting up to $\sim$ 7% less rain than running. Outside this limited wind-component region the advantage of running can be sizable: for example, in the absence of any wind ($v_{tail} = v_{cross} = 0$), walking leads to collecting approximately 37% more water than running.

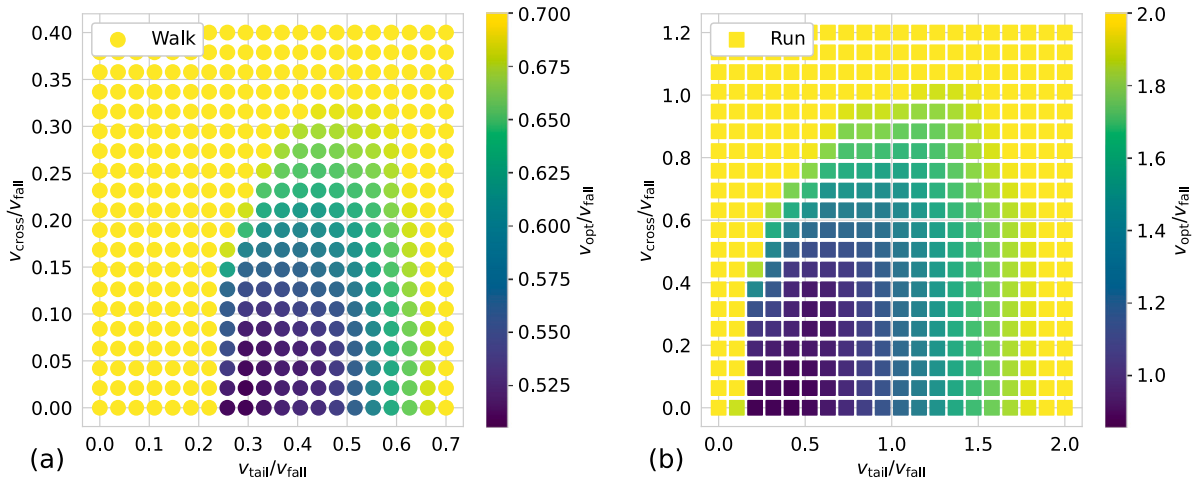

Fig. 12. Numerically evaluated optimal speed v_{opt} which minimizes the amount of collected water by the walking body, as a function of the horizontal wind components v_{tail} and v_{cross}. (a): walking-body model. (b): running-body model.

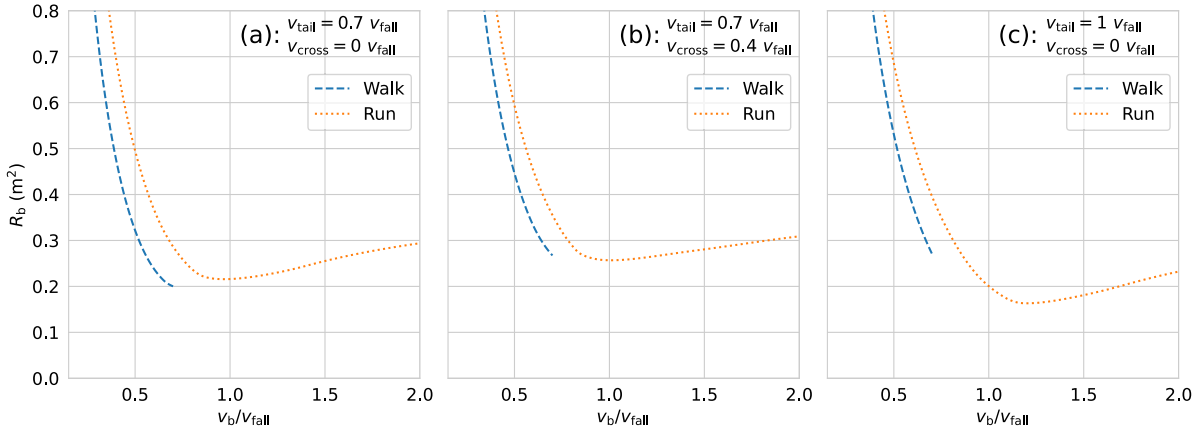

Fig. 13. Comparison of the wetness R_b as a function of the advancement speed v_b for the walking body model and the running one. Individual panels report different (indicated) wind conditions v_{tail} and v_{cross}.

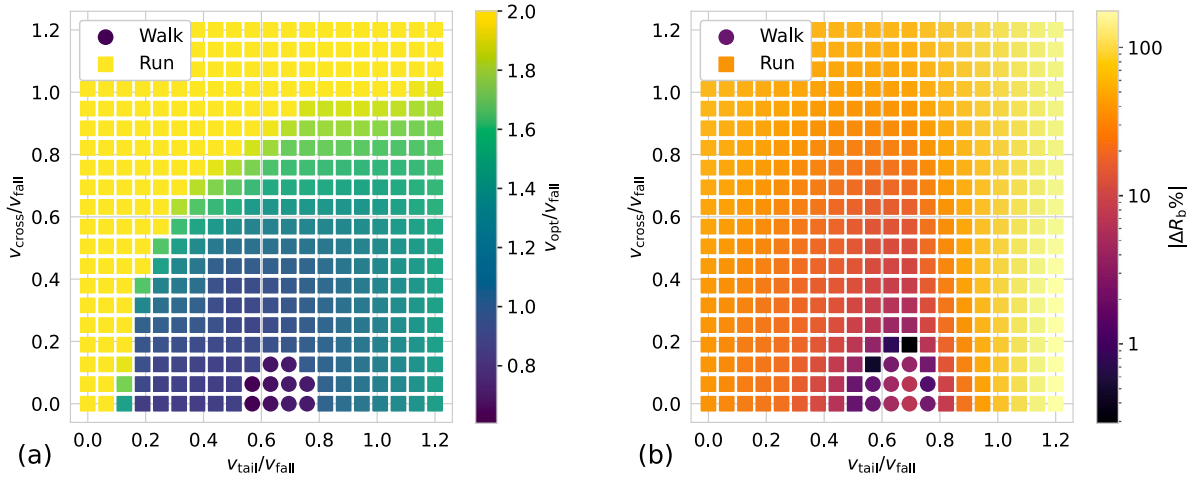

Fig. 14. (a): The optimum speed v_{opt} for the body model that achieves the smallest wetness between, as a function of the horizontal wind components v_{tail} and v_{cross}. (b): The relative advantage of running against walking $|\Delta R_b\%|$, Eq. (25), as a function of v_{tail} and v_{cross}. Squares: $\Delta R_b\% \geq 0$, where running is more advantageous than walking. Circles: $\Delta R_b\% < 0$, where walking is preferable to running.

To identify the reasons why under certain conditions that walking beats running, we evaluate the average projections of the walking and running bodies on the planes perpendicular to the three Cartesian axes x, y and z: we report these projections in Table 2. We see that, while the projected areas of the walking and running body models on the vertical planes are similar, the projected area on horizontal plane is

significantly smaller for the walking body than for the running body, due to the fact that the running body compared to the walking one leans the trunk forward, keeps the forearms more parallel to the ground, and swings its legs more widely, see Figs. 5 and 6. This difference leads to an advantage for the walking body when the relative rain velocity $\mathbf{v}_{\rm rel}$ is close to vertical, i.e., when $v_{\rm b} \simeq v_{\rm tail}$ and $v_{\rm cross} \simeq 0$. This projected-area advantage explains why precisely as the velocity approaches $v_{\rm tail}$ the walking body intercepts less rain than the running one, as long as the cross term remains negligibly small. Since the optimal velocity of the walking body assumes values in the $[0.5\ v_{\rm fall}, 0.7\ v_{\rm fall}]$ range, the values of $v_{\rm tail}$ which most favor the walking body are those in that same range, consistently with the circles reported in Fig. 14.

4. Conclusions

In the present paper, we construct and solve a relatively simple but accurate model for a human body advancing in a rain falling under steady-wind conditions. We use this model to identify the optimal advancement velocity that should be adopted with the aim of collecting as little water as possible. We tune our human-body model to simulate either a walking person and a running person. The model, although relatively simple, is fairly realistic, flexible, and systematically improvable, because it is based on a set of elementary shapes (spheres, parallelepipeds, and capsules) in relative motion.

We implement the model in a numerical code which simulates the movements of the individual body parts, and evaluates the amount of intercepted water over a given path, for arbitrary velocity of the wind and of the body. We check the convergence of the model as a function of the parameters that define the space and time resolution. As a function of the space and time discretizations, we evaluate the numerical accuracy of this method by comparing the deviation of the numerical results from the analytical expressions available for the sphere, parallelepiped, and capsule.

We compare our results with the ones provided by the traditional modeling based on a sphere or a parallelepiped, and find significant differences: the sphere modeling predicted the optimal velocity to always exist in the presence of a tailwind, and always exceed twice the falling speed of the rain; the parallelepiped modeling predicted the optimal velocity to always coincide with the tailwind (if present) and to be insensitive to any crosswind component. In contrast, our model shows that the optimal speed sits in between the two values and that it increases as the crosswind increases.

The general result is that in the absence of a tailwind it is better to run as fast as possible, while in the presence of a strong enough tailwind ($\gtrsim 20\%$ of the falling speed of the rain) there arises a nontrivial optimal walking/running velocity, that makes further speeding detrimental. The presence of a crosswind raises this optimal speed, and may even push it to the running range limit, if this side component is large enough. Notably, for a limited range of horizontal wind components, it is even preferable to walk than to run.

The developed code can be used without modifications to evaluate the soaking amount, and the corresponding optimum speed, for arbitrary shape-changing 3D objects, such as non-human animals, robots, or vehicles. One just needs to provide a file containing the geometric definition of the model for the target 3D object in terms of a suitable number of the implemented elementary shapes, namely spheres, parallelepipeds, and capsules.

The model could be further refined for a more accurate modeling of the human body and its dynamics. A study of body-shape effects on the optimal velocity could be instructive. It will also be intriguing to verify how the results would be affected by the introduction of an umbrella that partly shields the body from the rain, as was done for simple geometric shapes by T. Echekki [9].

CRediT authorship contribution statement

Cristian Angelo Crespi: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Nicola Manini:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge the support of the APC central fund of the university of Milan.

Appendix A. Technical implementation

The developed code is available on GitHub [20]. The main numerical code is written in C++; for data analysis and visualization we adopt Python. This Appendix focuses on the main C++ code.

We start by introducing the general organization of the code before delving into the main details of the algorithm. As discussed in Section 2, the primary objective is the evaluation of $R_{\rm b}$ for a complex body composed of elementary shapes. This task requires the evaluation of the projection of a complex body on a plane perpendicular to the relative rain velocity. As detailed in Appendices A.1 and A.2, for this evaluation the code generates a portion of this plane, projects the elementary shapes forming the full body, and evaluates the total projected area. The implemented method avoids double counting of overlapping projections. In Appendix A.2 we describe the implementation of the relative motion of the connected elementary bodies, which simulate the dynamic model of a walking/running human being.

In the following, 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.

A.1. Projection surface

The numerical evaluation of the 2D projection of a 3D body relies on an array of straight lines parallel to $\mathbf{v}_{\rm rel}$, i.e. perpendicular to the projection plane. These lines, which we call rays, are distributed in space on a uniform grid. To evaluate the projected area, we count how many rays intersect the body. Each ray can be parameterized as follows:

$$\mathbf{R}(\tau) = \mathbf{R}_0 + \tau \mathbf{v}_{\rm rel}, \quad \tau \in \mathbb{R}. \tag{A.1}$$

We call $\mathbf{R}_0$ the "ray origin". In the code, a ray is represented by an object of the `Ray` class. Its data members are the origin vector `R0` and a double `Weight` between 0 and 1 that describes how close to the body edge the `Ray` hits. `Ray` also contains a static data member: the vector `V` representing $\mathbf{v}_{\rm rel}$.

We implement the projection by means of the class `ProjSurf`. The data members of `ProjSurf` are: a vector `rays` of `Ray` objects, a double `dx` representing the spacing of the array of rays, and a matrix `H`. `H[i]`, with `i` $= 1, \ldots, 6$, contains the coordinates of the vertices of the hexagonal projection of the simulation box, which is a parallelepiped with sides parallel to the axes and a vertex positioned at the origin that is assumed to fully contain the body we project. `H[0]` stores the coordinates of a vertex of the box that is inside the projection surface.

The `ProjSurf` constructor takes as arguments a vector `box`, a vector `vel` representing $\mathbf{v}_{\rm rel}$ (used to set the corresponding static member `V` of `Ray`), and a double `Dx` used to initialize the member data `dx`.

box represents the lengths of the three sides of the simulation box. The ProjSurf constructor first uses the function FindMiddle to find the vertex of the box shared by the faces of the box exposed to rain, and assigns that vertex to H[0]. Then it calls a function FindHexProj to compute the projection of the vertices of the "wet faces" onto a plane perpendicular to vel and passing through H[0]. The resulting points are assigned to the remaining elements of H.

This hexagon whose corners are stored in H is then filled with a regular array of ray origins. To do this we generate two perpendicular unit vectors u1 and u2 laying on the projection plane. For u1 we take the longest vector between H[i] − H[0] with $i = 1, \dots, 6$, then normalized to unit length. u2 is then obtained by normalizing the cross product u1 × vel, which is perpendicular to both u1 and vel by construction. Then the maximum projections $\texttt{maxu}\alpha = |(\texttt{H[i]} - \texttt{H[0]}) \cdot \texttt{u}\alpha|$ along the unit vectors uα are computed for $\alpha = 1, 2$. With this definition, the hexagon defined by H is contained in the rectangle centered at H[0] whose sides are the vectors 2*maxu1*u1 and 2*maxu2*u2. We generate a square lattice with spacing dx along u1 and u2 inside this rectangle. The location of each generated point is checked for laying inside the hexagon: when it is, the Ray constructor builds a Ray object with that point as its origin R0. This Ray object is then added to the list rays. The bool function PointIsInsideT verifies whether a point lays inside the hexagon. This function considers the 4 triangles with vertices H[1], H[i] and H[i+1] (with $i = 2, 3, 4, 5$) that compose the hexagon. Then PointIsInsideT uses barycentric coordinates to determine if the point is inside each triangle, following the ray-tracing recipe of section 3.2 of Ref. [11].

Once the ProjSurf constructor has completed its tasks, it is possible to evaluate the projection of a Body-type object (for details, see Appendices A.2 and A.2 below). This is achieved by means of the function BodyProj, which returns the area of the projection of the input Body on the projection plane. To do this, on each Ray in rays BodyProj calls the Body method Check, which returns a weight between 0 and 1 that describes how close the Ray is to the Body, and assigns this number to the Weight of the ray. In the end, BodyProj returns the sum of all these weights multiplied by $\texttt{dx}^2$, namely the area of a primitive cell of the square lattice of ray origins. In detail, Check evaluates the signed distance delta_r between the Ray and the outer edge of the Body. This distance is positive for a Ray which does not intersect the Body, and negative when it does. Check then returns the result a "weight" resulting from the application of a function d_to_w to delta_r. d_to_w is a monotonically-decreasing continuously-differentiable function of 1 numeric variable delta_r, which returns a weight that decreases from 1 for $\texttt{delta\_r} < -\texttt{dx}$ to 0 for $\texttt{delta\_r} > \texttt{dx}$. The adoption of this smooth edge ensures that, despite being based on a space discretization, BodyProj is a continuous functions of both the position of the Body and the orientation of the projection plane.

A.2. Bodies

We define a parent class Body from which all the classes representing the individual shapes inherit. We first discuss the Body methods Prime and Check. Each derived class comes with its own implementation of these methods that overrides the general one. The method Prime prepares the Body to be checked, and Check takes as argument a Ray and returns a weight between 0 and 1 that describes how well the Ray intersects the Body. We divide the checking process into Prime and Check because an efficient way of evaluating the distance of the Ray from the edges of the body is to project the body on the projection plane and then evaluate the distance of the R0 of the ray from the projection of the body. Since the code needs to check a large number of rays it is more efficient to first project the body by calling Prime once, and then call Check for each ray without projecting the body each time. The function Prime thus groups all preliminary operations that 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 the projection plane. As discussed in Section 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 evaluates the projection of cent onto the projection plane and stores it in Hcent. When called on a Ray, Check evaluates delta_r as the distance between R0 and Hcent, minus rad.

Parallelepiped. We implement parallelepipeds through the Parallelepiped class, derived 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 vector $\mathbf{s}_i$, and a matrix H with the same content and meaning as the H data member of ProjSurf. Prime projects the vertices of the wet faces of the parallelepiped onto the projection surface, with H[0] storing the projection of the vertex shared by all the wet faces, and H[i] storing the vertices of the hexagonal projection. Check evaluates delta_r by calculating the distance between R0 and each side of H using PointSegDist, providing the smallest of the resulting values, and using PointIsInsideT to determine whether R0 is inside of H to assign the appropriate sign to delta_r.

Capsule. We implement capsules through the Capsule class, derived from Body. Its additional data members are two vectors: l1 and l2, representing the position of the two endpoints of the capsule axis, a double rad representing the capsule radius and two vectors, H1 and H2 representing the projections of l1 and l2 on the ProjSurf. As discussed in Section 2.1.2 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 plane and assigns them to H1 and H2. Check evaluates delta_r as the difference between the distance between R0 and the segment defined by the two endpoints H1 and H2, and rad.

Composite object. We implement a complex body consisting of a collection of 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 pointed at by bodies, preparing all of them to be checked. When Check is called on a Ray, it calls Check for all the elements of bodies and returns the highest value found this way. If, for a given ray, the evaluation of Check on an element of bodies results in a value of 1, indicating full contact, then the method returns 1 immediately, skipping further Check evaluations on the remaining bodies.

We now detail our implementation of the relative movements of the individual body element collected in ManyBody. We need our body elements to move 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 a second Body B, we say that B is a sub-body of A, and A is the parent body of B. A sub-body moves relative to the frame of reference of its parent body. As a result, when the parent body moves, so do all its sub-bodies. For example, in the walking human model the Capsule representing the forearm is a sub-body of the Capsule representing the upper arm: as a result, when the upper arm swings back and forth, this movement propagates to its forearm.

We model the movement of each body element as a translation followed by a rotation around an axis. The Body class includes a vector data member rotcent representing the center of rotation, and a vector data member rotax that stores the rotation axis. Translations are implemented by the Body method Translate, which translates

the Body (in practice all its defining points, including its rotcent) by a vector shift which Translate takes as argument.

Rotations are implemented via the Body method Rotate, which takes the center of rotation rot0 and a rotation matrix rotmat as arguments. Applying this rotation matrix directly is sufficient to rotate vectors that join two points in space, e.g. the sides of a parallelepiped. In contrast, for vectors storing absolute positions in space, e.g. the center of a sphere body, we use the function RotatePoint, which takes the rotation center rot0 into consideration. Both Translate and Rotate are suitably overridden in each class derived from Body.

We call $\mathbf{T_r}(t)$ and $\Theta(t)$ respectively the translation and the angle of the rotation that the body is subject to at time t. Since we want our movements to be periodic we write both $\Theta(t)$ and $\mathbf{T_r}(t)$ as a discrete Fourier series, which we truncate to contain a finite number of components. These Fourier components are stored in the Body data members trans (a matrix) and w (a vector), while the phase terms are stored in transPhi and wPhi, and the constant terms in delta0 and theta0. $\mathbf{T_r}(t)$ is then calculated as:

$$\mathbf{T_r}(t) = \texttt{delta0} + \sum_{\texttt{i}=0}^{i_{\max}} \texttt{trans[i]} \sin\left(2\pi\,(\texttt{i}+1)\,\frac{t}{T} + \texttt{transPhi[i]}\right)\,, \quad \text{with} \quad i_{\max} = \texttt{trans.size()} - 1. \tag{A.2}$$

Here the sum is executed only if the upper limit exceeds the lower one.

The instantaneous limb rotation angle $\Theta(t)$ is similarly evaluated as:

$$\Theta(t) = \texttt{theta0} + \sum_{\texttt{i}=0}^{i_{\max}} \texttt{w[i]} \sin\left(2\pi\,(\texttt{i}+1)\,\frac{t}{T} + \texttt{wPhi[i]}\right)\,, \quad \text{with} \quad i_{\max} = \texttt{w.size()} - 1. \tag{A.3}$$

$\mathbf{T_r}(t)$ and $\Theta(t)$ are implemented in the code by the Body methods getDelta and getTheta, which take as argument a quantity t representing the ratio t/T. The current values of $\mathbf{T_r}(t)$ and $\Theta(t)$ are stored in Body as data members delta and theta. They are initialized to 0 by the constructor.

We now have all the elements needed to properly describe the Move method. The following implementation of Move is shared by all Body derived objects, except ManyBody. When called with a double tNew as argument, Move uses getDelta to evaluate deltaNew, and calculates the step translation deltaShift that brings the current body position (at time t) to its new position at time tNew: deltaShift = deltaNew − delta. Similarly, Move also evaluates thetaNew and the incremental rotation angle thetaShift that rotates the body to its new orientation: thetaShift = thetaNew − theta. After computing deltaShift and thetaShift, Move updates delta and theta to their new values deltaNew and thetaNew, respectively. Given thetaShift, the function RotMat is called taking thetaShift and rotax as argument, generating the required rotation matrix rotmat. At this point, Move applies the translation (using Translate) and rotation (with Rotate) sequence to the body. In the end, Move calls the Body method BeMoved on all its sub-bodies. BeMoved takes trans, rotmat, and rotcent as input, and moves the sub-body accordingly. This includes rotating the rotcent and rotax of the sub-body, and rotating all vectors in its trans. In this way, the system of reference in which the sub-body moves is subjected to the same rototranslation as its parent body. This procedure works recursively, so that BeMoved is called on the sub-body's own sub-bodies (if any) ensuring that they are moved correctly, too.

For the special case of a ManyBody, Move calls the Move method sequentially on all elements stored in bodies. BodyProj computes the average projection of a dynamic body during motion, when it is called with three extra arguments: two double tmin and tmax, plus an unsigned int nstep. BodyProj generates motion snapshots of the Body at nstep equally-spaced time steps between tmin and tmax, which will typically cover one full period of motion by setting tmin = 0 and tmax = 1. Note that the physical time step is $dt = T/\texttt{nstep}$. At each time step BodyProj evaluates the weights of the rays as detailed in Appendix A.1. At the end, BodyProj returns the nstep-averaged total weight multiplied by $\texttt{dx}^2$.

A.3. The simulation box

To determine a simulation box that fully contains the Body at every step of its periodic deformation, we use the Body method GetBox. This method makes use of the auxiliary Body method GetBounds, which, when called with arguments tmin, tmax, and nstep, returns two vectors, low and high, that contain respectively the lowest and highest x, y and z coordinates that fully contain the body at every time step; for example if low[0] = 0.2 and high[0] = 0.6, then at all time steps the body x coordinate is fully contained in the interval $[0.2, 0.6]$. The evaluated interval is also the smallest one that satisfies this condition. The simulation box is then determined based on the resulting low and high suitably incremented by a tolerance epsilon. GetBox then translates the Body by a vector −low, and returns as the simulation box the difference between the vectors high and low. In this way, a corner of the simulation box coincides with the origin.

A.4. Minimization

As anticipated in Section 3, the minimization of $R_b(v_b)$ consists in two steps: (i) a Brent minimization leading to a rough estimation of the minimum; (ii) the parabolic fitting of a relatively small number N_{fit} points scattered around this initial estimate of the minimum. The minimum of the parabola is adopted as the best numerical estimation of the absolute minimum of $R_b(v_b)$.

This minimization is implemented through the function MinFit, which takes the following parameters as input: the body, the maximum physically accessible body velocity vmax, the number nfit of points to be fitted, and the spacing dv between these points. MinFit returns a tuple containing four doubles representing v_{opt}, its errorbar, R_{min} and its errorbar, and additionally a matrix containing the evaluated points (illustrated by red dots in Fig. 8), with the values of v_b in the first column and the corresponding values of R_b in the second column.

The C++ function MinFit attempts to minimize $R_b(v_b)$. The first step taken by MinFit is bracketing the interval where the minimum is to be searched, i.e. identifying three points $a < b < c$ such that $f(b) < f(a)$ and $f(b) < f(c)$. For the lower bracket point, we take the small value a = dv, because $R_b(v_b)$ is undefined at $v_b = 0$. For the upper bracket point c = vmax is appropriate. MinFit then checks if b = vmax$-i$ dv satisfies the bracketing condition for $i = 1, 2, 3$, which should be the case if a minimum exists within the $[a, b]$ interval at all. If this bracketing method fails, then MinFit assumes that $f(x)$ is monotonically decreasing in our interval, and therefore it returns vmax and R_b(vmax). If bracketing succeeds, then MinFit uses Brent's minimization method to identify a first estimate of the minimum. Then it evaluates the function at nfit points spaced by dv around the Brent minimum. Since the function $R_b(v_b)$ suffers from numerical noise, local minima can trap Brent's minimization. As a result the nfit points might be disproportionately skewed at one side of the global minimum: this can lead to inaccurate fits, since a parabolic approximation only makes sense around the minimum. To prevent this occurrence, MinFit computes a preliminary parabolic fit and checks if nearly half of the nfit points sit at both sides of the resulting minimum xmin. If this test succeeds, MinFit terminates, returning xmin and the corresponding value of the function. If this test fails, e.g. just 2 points sit at the left of the estimated xmin, and 4 points sit at its right, then MinFit drops the point with the largest x and evaluates one extra point at $x = x_{\text{small}} - \texttt{dv}$, where x_{small} is the lowest point. The parabolic fit is then re-evaluated on the new set of nfit points. The entire test-fit procedure is then repeated until the condition for the minimum sitting at the middle of the set of points is satisfied, with a check to prevent the risk that numerical noise might generate an infinite loop.

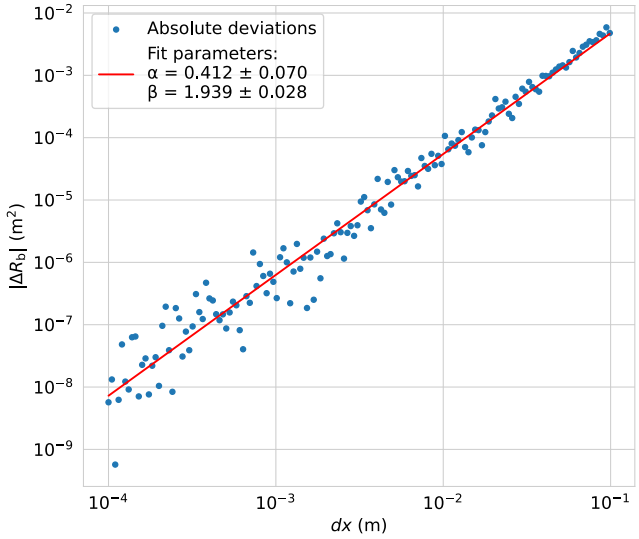

Fig. B.15. The absolute deviation $|\Delta R_b(dx)|$ as a function of the resolution dx of the array of probing rays, for a sphere with radius $r = 0.5$ m and velocity components $v_{tail} = 0.5\, v_{fall}$, $v_{cross} = 0.25\, v_{fall}$, and $v_b = 2\, v_{fall}$. The deviations (blue points) are fitted through Eq. (B.2) (red line), and the best-fit parameters are indicated. For a comparison providing the relative uncertainty, it is useful to report $R_b = 0.715$ m^2.

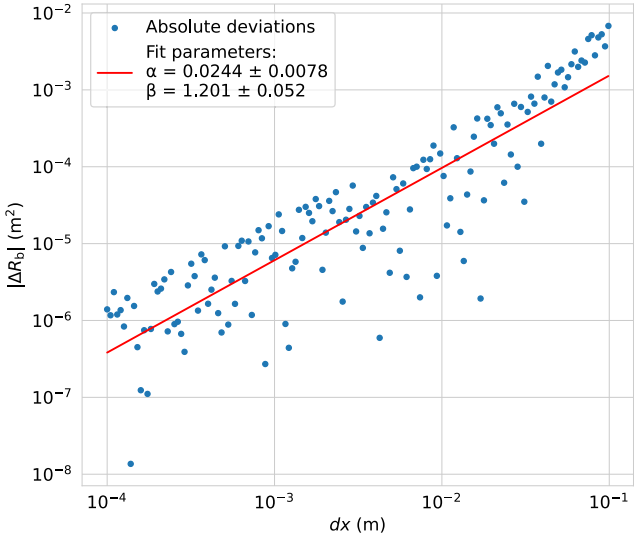

Fig. B.16. Same as Fig. B.15 for a parallelepiped with sides $s_x = 0.4$ m, $s_y = 0.6$ m, $s_z = 0.8$ m. Here, $R_b = 0.520$ m^2.

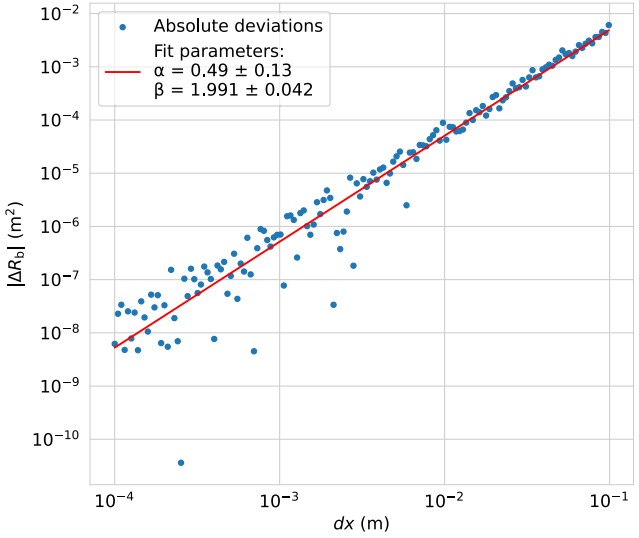

Fig. B.17. Same as Fig. B.15 for a capsule with a radius $r = 0.3$ m and $\Delta\mathbf{l} = (0.4, 0.4, 0.4)$ m. Here, $R_b = 0.522$ m^2.

Appendix B. Code validation

To validate the code and estimate the numeric uncertainty, we compare the numerical results of `BodyProj`, as described in Appendix A, with the analytic formulas of Section 2 for a sphere, a parallelepiped, and a capsule. With $\tilde{R}_b(dx)$ we indicate the estimation of R_b evaluated with our code when a spatial resolution dx is adopted. The deviation from the exact analytic result is then:

$$\Delta R_b(dx) = \tilde{R}_b(dx) - R_b. \tag{B.1}$$

For example, Fig. B.15 reports the absolute value of this deviation computed for a sphere with radius $r = 0.5$ m and velocity components $v_{tail} = 0.5\, v_{fall}$, $v_{cross} = 0.25\, v_{fall}$, $v_b = 2\, v_{fall}$, and numerous values of dx. The log–log plot suggests that $|\Delta R_b(dx)|$ approximately follows a power law:

$$|\Delta R_b(dx)| = \alpha\ dx^{\beta}, \tag{B.2}$$

where α and β are fit parameters. To fit the absolute deviations to this power law, we pass to the logarithms:

$$\ln(|\Delta R_b(dx)|) = \ln(\alpha) + \beta \ln(dx)\,, \tag{B.3}$$

and evaluate α and β by means of a linear regression. The resulting best fit is reported as a line in Fig. B.15, showing that the data are consistent with power-law overall trends.

We repeat this test, with the same velocity components, for a parallelepiped with sides $s_x = 0.4$ m, $s_y = 0.6$ m, $s_z = 0.8$ m (Fig. B.16), and for a capsule with radius $r = 0.3$ m and axis $\Delta\mathbf{l} = (0.4, 0.4, 0.4)$ m (Fig. B.17). For all these body elements we find that the numerical deviation does decrease as a power law of the spatial resolution, Eq. (B.2). We also note that the parallelepiped exhibits a noticeably slower convergence ($\beta \simeq 1$) than both the sphere and the capsule ($\beta \simeq 2$). This is likely due to the sharp edges and vertices of the parallelepiped, which do not affect the rounded shapes instead.

Figs. B.18, B.19, and B.20 report a comparison of $R_b(v_b)$ and $\tilde{R}_b(v_b, dx)$ as a function of v_b for the same three considered elementary bodies, and the same wind components. The calculations are done for $dx = 0.001$ m. Clearly this spatial resolution provides a satisfactory precision, certainly sufficient to determine the minimum of $R_b(v_b)$, if any. Figs. B.18, B.19, and B.20 also illustrate common situations where the body shape determines the existence and value of an optimal speed.

We also need to test the code's ability to handle rain shadowing in composite bodies correctly. To do so we consider a complex body consisting of two parallelograms: p_1 with sides $s_x = 0.8$ m, $s_y = 0.5$ m, $s_z = 0.8$ m, and a smaller p_2 with sides $s_x = 0.6$ m, $s_y = 0.3$ m, $s_z = 0.6$ 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 $v_{cross} = 0\, v_{fall}$. 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

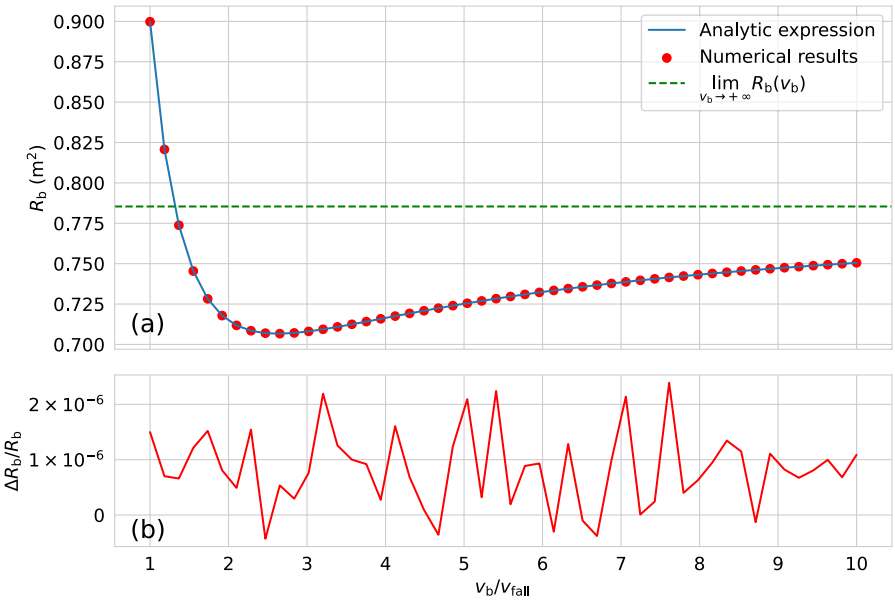

Fig. B.18. (a) Comparison of the numeric estimate $\tilde{R}_{(b)}(v_b, dx)$ of the wetness (dots) with its exact value $R_b(v_b)$ (solid) for the sphere of Fig. B.15, as a function of its speed v_b. Dashed line: the infinite-speed limit, equal to the sphere cross section πr^2. (b) The relative deviation $\Delta R_b(dx)/R_b(dx)$ between the numerical and the exact evaluations. For the numerical calculations we adopt a spatial resolution $dx = 0.001$ m. The velocity components $v_{tail} = 0.5\, v_{fall}$ and $v_{cross} = 0.25\, v_{fall}$.

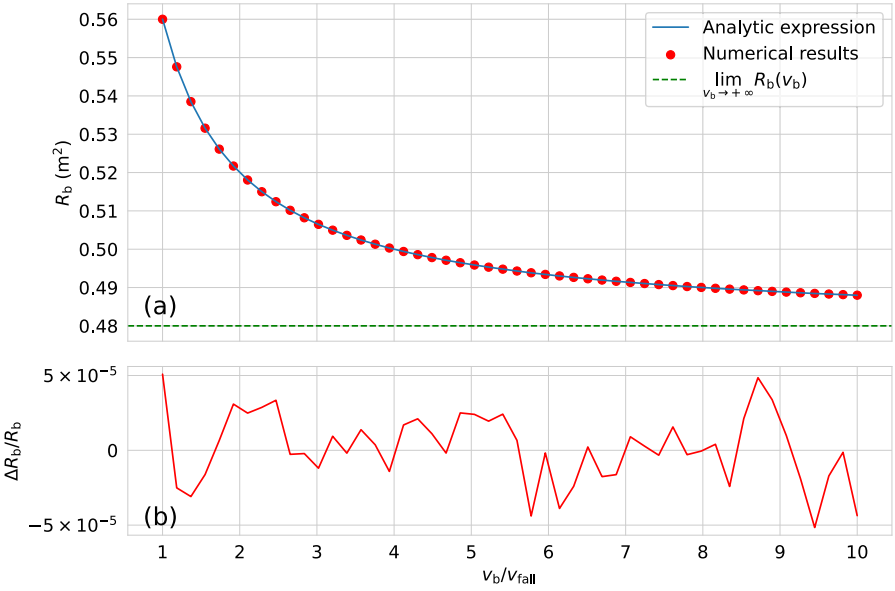

Fig. B.19. Same as Fig. B.18 for a parallelepiped with sides $s_x = 0.4$ m, $s_y = 0.6$ m, $s_z = 0.8$ m.

fully outside and not overshadowed by p_1. Furthermore since $\mathbf{v}_{rel}$ has no y component, $p1$ and $p2$ do not overshadow one another as soon as they are no longer in contact, which occurs for $|\mathbf{c}_2 - \mathbf{c}_1| > 0.4$ m, where we can predict that $R_{2p} = R_{p1} + R_{p2}$. The numerical results reported in Fig. B.21 confirm that R_{2p} behaves exactly as expected.

Appendix C. Error analysis for the dynamical body

To estimate the error on the numerical estimation of the wetness $\tilde{R}_b$ of dynamically evolving bodies, we face two additional challenges compared to the static elementary bodies considered in Appendix B. First, we no longer have access to an analytic solution with which the numerical results can be compared. Secondly, we now have two simultaneous sources of numerical error: the finite spatial resolution dx of the rays, and the time discretization dt.

We assume that both discretization errors decay as independent power laws for small dx and dt, thus allowing us to discuss the two sources of errors separately. We express the numerically evaluated wetness $\tilde{R}_b(dx, dt)$ approximately as:

$$\tilde{R}_b(dx, dt) = R_b + E_{dx}(dx) + E_{dt}(dt), \tag{C.1}$$

$$E_{dx}(dx) \simeq A_{dx}\, dx^{B_{dx}}, \tag{C.2}$$

$$E_{dt}(dt) \simeq A_{dt}\, dt^{B_{dt}}. \tag{C.3}$$

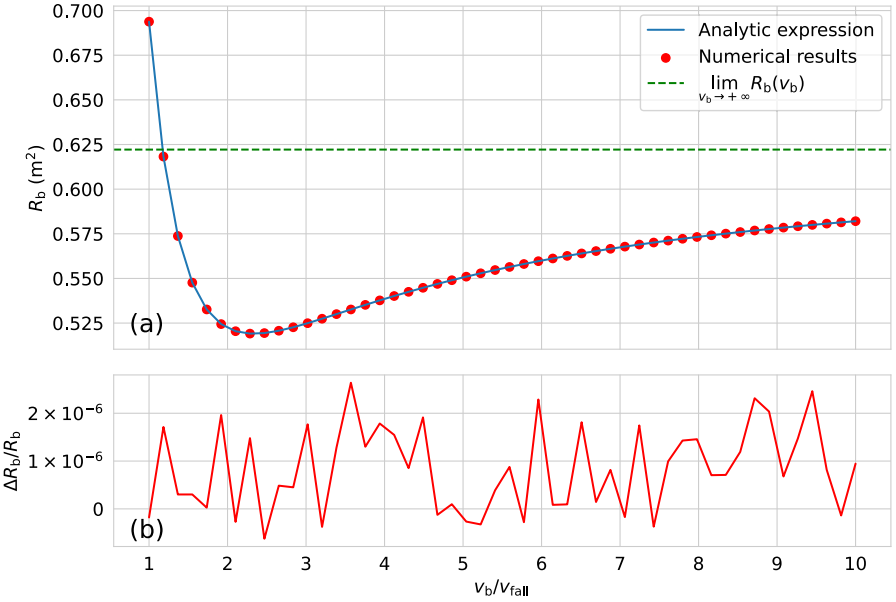

Fig. B.20. Same as Fig. B.18, but for a capsule with radius $r = 0.3$ m and $\Delta\mathbf{l} = (0.4, 0.4, 0.4)$ m.

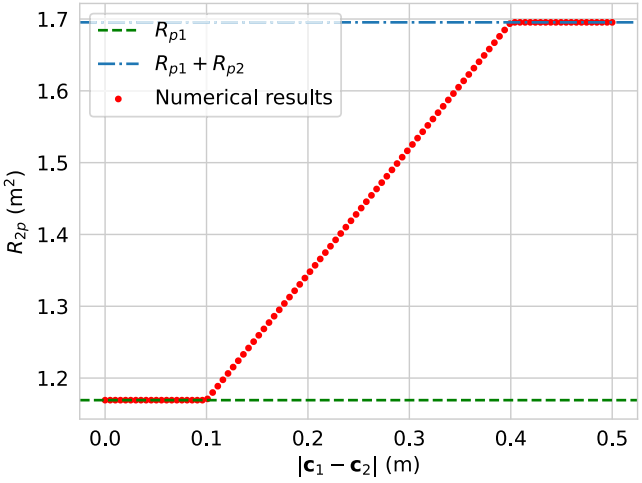

Fig. B.21. Wetness R_{2_p} of a complex body consisting of two parallelepipeds p_1 and p_2 of different size (details in the text) as a function of the distance between their centers $|\mathbf{c}_1 - \mathbf{c}_2|$. The calculation is carried out with $dx = 0.001$ m, $v_b = 0.52\, v_{fall}$, $v_{tail} = 0$ and $v_{cross} = 0$.

Here E_{dx} and E_{dt} are the discretization errors resulting from dx and dt respectively. We adopt 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 determine an estimation of $R_b + E_{dx}$ and its error. Given this estimation of the $dt \to 0$ asymptotic value, we proceed to evaluate the absolute deviations following a procedure similar to that of Appendix B:

$$\left|\Delta R_b(dt)\right| = \left|\tilde{R}_b(dx, dt) - (R_b + E_{dx})\right|. \tag{C.4}$$

We can then fit these absolute deviations with a power law:

$$\left|\Delta R_b(dt)\right| = \alpha_{dt}\ dx^{\beta_{dt}} \tag{C.5}$$

In the investigation of the numerical error due to the finite spatial resolution dx we repeat the procedure, but with dt and dx exchanged. Figs. C.22 and C.23 report the results of this error analysis for the walking body on dt and dx respectively, along with the adopted conditions for these simulations. We have carried out the same analysis for the running body, with very similar results.

The error associated to dx behaves similarly to that investigated in Appendix B for elementary solids. In comparison, deviations due to dt are generally smaller, and exhibit a faster power-law convergence to zero. We should note that the bulk of the computational cost of the adopted algorithm stems from ray checking. The number of rays, and therefore the number of checks, grows proportionally to dx^{-2}. In contrast, the number of checks is linear in $\texttt{nstep} \propto dt^{-1}$. As a result, it is computationally far cheaper to decrease dt than dx, if one wishes to reduce the associated error.

Data availability

The code is publicly available on GitHub at https://github.com/Cr3sp1/RainSimulation.

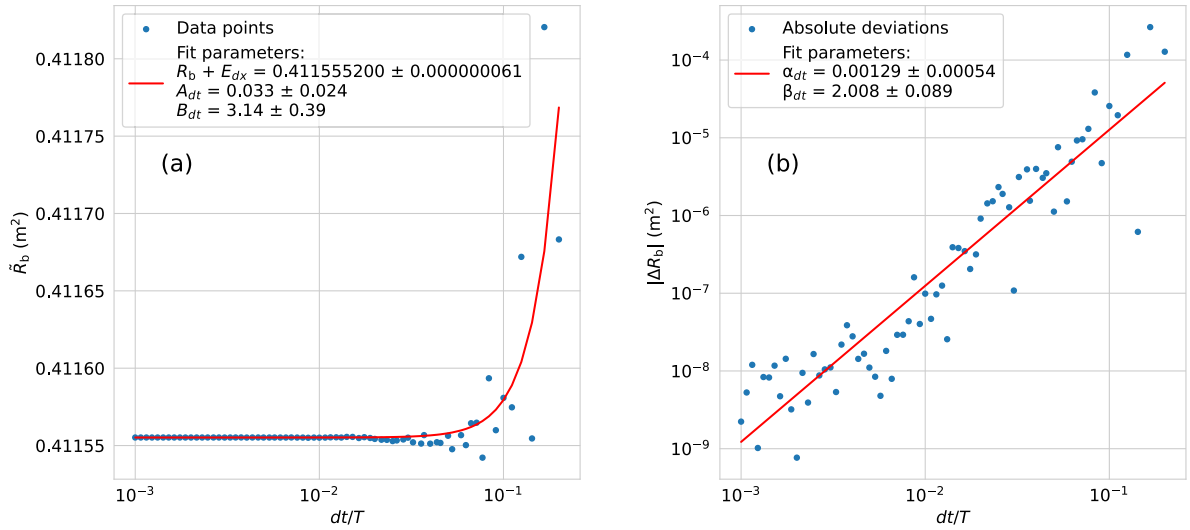

Fig. C.22. Analysis of the error associated to the finite discretization time step dt for the walking body. (a) A scaling of the wetness $\tilde{R}_w$ as a function of the time step dt. Blue dots: numeric data points; red line: the best-fit power law, Eq. (C.3). (b) Blue points: absolute deviations $|\Delta R_b(dt)|$; red line: the best-fit power law, Eq. (C.5). In these calculations we take $dx = 0.001$ m, $v_{tail} = 0.5\ v_{fall}$, $v_{cross} = 0.25\ v_{fall}$ and $v_b = 2\ v_{fall}$.

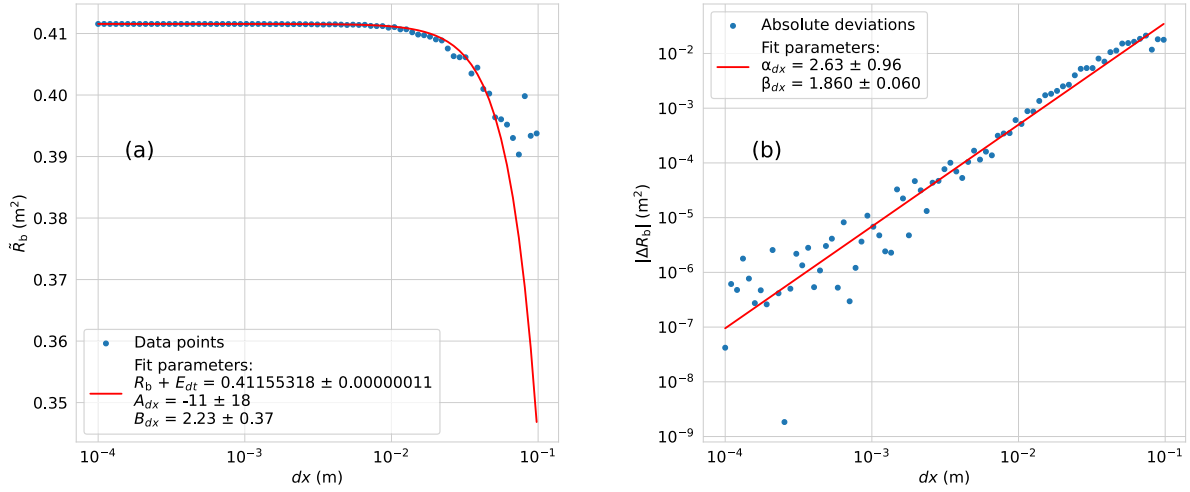

Fig. C.23. Analysis of the error associated to the finite spatial resolution dx, similar to Fig. C.22, but with the quantities expressed as functions of dx. The time resolution is fixed at $dt = 0.02\,T$ and the other parameters are the same.

References

[1] B. Schwartz, M. Deakin, Walking in the rain, reconsidered, Math. Mag. 46 (1973) 272–276, http://dx.doi.org/10.2307/2688265.

[2] D. Hailman, B. Torrents, Keeping dry: The mathematics of running in the rain, Math. Mag. 82 (2009) 266–277, http://dx.doi.org/10.4169/193009809X468698.

[3] F. Bocci, Whether or not to run in the rain, Eur. J. Phys. 33 (2012) 1321, http://dx.doi.org/10.1088/0143-0807/33/5/1321.

[4] T. Kroetz, The "running in the rain" problem revisited: an analytical and numerical approach, Rev. Bras. Ensino Fis. 31 (2009) 4304–4309, http://dx.doi.org/10.1590/S1806-11172009000400006.

[5] A.D. Angelis, Is it really worth running in the rain? Eur. J. Phys. 8 (1987) 201, http://dx.doi.org/10.1088/0143-0807/8/3/011.

[6] J.J. Holden, S.E. Belcher, A. Horvath, I. Pytharoulis, Raindrops keep falling on my head, Weather 50 (1995) 367–370, http://dx.doi.org/10.1002/j.1477-8696.1995.tb07246.x.

[7] S.A. Stern, An optimal speed for traversing a constant rain, Am. J. Phys. 51 (1983) 815–818, http://dx.doi.org/10.1119/1.13124.

[8] T.C. Peterson, T.W.R. Wallis, Running in the rain, Weather 52 (1997) 93–96, http://dx.doi.org/10.1002/j.1477-8696.1997.tb06281.x.

[9] T. Echekki, In the rain with and without an umbrella? The Reynolds transport theorem to the rescue, Eur. J. Phys. 41 (2019) 1, http://dx.doi.org/10.1088/1361-6404/ab4b62.

[10] S. Hartel, C. Faber, From differentiable modeling to system optimization: the full journey using geometric algebra exemplarily applied to improve light sectioning systems, TM. Tech. Mess. (2026) http://dx.doi.org/10.1515/teme-2025-0127.

[11] A.S. Nery, N. Nedjah, F.M. França, An efficient parallel architecture for ray-tracing, Analog. Integr. Circ. Sig. Process. 70 (2012) 189–202, http://dx.doi.org/10.1007/s10470-011-9739-x.

[12] M. Zaegel, M. Vehils-Vinals, H. Guastalla, B. Benabou, A. Gires, Should you walk, run or sprint in the rain to get less wet? Eur. J. Phys. 45 (2023) 025802, http://dx.doi.org/10.1088/1361-6404/ad06bf.

[13] D.C. Lay, S.R. Lay, J.J. McDonald, Linear Algebra and Its Applications, Global Edition, Pearson Education Limited, Harlow, Essex, England, 2016.

[14] L. Pournin, M. Weber, M. Tsukahara, J.-A. Ferrez, M. Ramaioli, T.M. Liebling, Three-dimensional distinct element simulation of spherocylinder crystallization, Granul. Matter 7 (2005) 119–126, http://dx.doi.org/10.1007/s10035-004-0188-4.

[15] R. Schneider, Convex Bodies: the Brunn–Minkowski Theory, Cambridge University Press, Cambridge, UK, 1993.

[16] P. Huang, S. Pan, Y. Yang, Positive center sets of convex curves, Discrete Comput. Geom. 54 (2015) 728–740, http://dx.doi.org/10.1007/s00454-015-9715-9.

[17] A. Keys, F. Fidanza, M.J. Karvonen, N. Kimura, H.L. Taylor, Indices of relative weight and obesity, J. Chronic Dis. 25 (1972) 329–343, http://dx.doi.org/10.1016/0021-9681(72)90027-6.

[18] B. Bogin, M.I. Varela-Silva, Leg length, body proportion, and health: A review with a note on beauty, Int. J. Env. Res. Public Health 7 (2010) 1047–1075, http://dx.doi.org/10.3390/ijerph7031047.

[19] L. da Vinci, Uomo vitruviano, 1490, photography by Luc Viatour, https://en.wikipedia.org/wiki/File:Da_Vinci_Vitruve_Luc_Viatour.jpg.
[20] C.A. Crespi, Rain simulation, 2026, https://github.com/Cr3sp1/RainSimulation.
[21] P.R. Cavanagh, The biomechanics of lower extremity action in distance running, J. Foot Ankle 7 (1987) 197–217, http://dx.doi.org/10.1177/107110078700700402.
[22] A.K. Yegian, Y. Tucker, S. Gillinov, D.E. Lieberman, Straight arm walking, bent arm running: gait-specific elbow angles, J. Exp. Biol. 222 (2019) http://dx.doi.org/10.1242/jeb.197228.
[23] A. Dos Santos, T.H. Nakagawa, G.Y. Nakashima, C.D. Maciel, F. Serrão, The effects of forefoot striking, increasing step rate, and forward trunk lean running on trunk and lower limb kinematics and comfort, Int. J. Sport. Med. 37 (2016) 369–373, http://dx.doi.org/10.1055/s-0035-1564173.
[24] V. Bringi, M. Thurai, D. Baumgardner, Raindrop fall velocities from an optical array probe and 2-D video disdrometer, Atmos. Meas. Tech. 11 (2018) 1377–1384, http://dx.doi.org/10.5194/amt-11-1377-2018.
[25] V.L. Billat, A. Demarle, J. Slawinski, M. Paiva, J.-P. Koralsztein, Physical and training characteristics of top-class marathon runners, Med. Sci. Sports Exerc. 33 (2001) 2089–2097, http://dx.doi.org/10.1097/00005768-200112000-00018.
[26] R.L. Waters, B.R. Lunsford, J. Perry, R. Byrd, Energy-speed relationship of walking: standard tables, J. Orthop. Res. 6 (1988) 215–222, http://dx.doi.org/10.1002/jor.1100060208.
[27] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing, Cambridge University Press, Cambridge, UK, 2007.