Artificial Intelligence In Medicine 98 (2019) 27–34

Contents lists available at ScienceDirect

Artificial Intelligence In Medicine

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

Deep multiphysics: Coupling discrete multiphysics with machine learning to attain self-learning in-silico models replicating human physiology

Alessio Alexiadis

School of Chemical Engineering, University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom

ARTICLE INFO

Keywords:
Discrete multiphysics
Reinforcement Learning
Coupling first-principles models with machine learning
Particle-based computational methods

ABSTRACT

Objectives: The objective of this study is to devise a modelling strategy for attaining in-silico models replicating human physiology and, in particular, the activity of the autonomic nervous system.
Method: Discrete Multiphysics (a multiphysics modelling technique) and Reinforcement Learning (a Machine Learning algorithm) are combined to achieve an in-silico model with the ability of self-learning and replicating feedback loops occurring in human physiology. Computational particles, used in Discrete Multiphysics to model biological systems, are associated to (computational) neurons: Reinforcement Learning trains these neurons to behave like they would in real biological systems.
Results: As benchmark/validation, we use the case of peristalsis in the oesophagus. Results show that the in-silico model effectively learns by itself how to propel the bolus in the oesophagus.
Conclusions: The combination of first principles modelling (e.g. multiphysics) and machine learning (e.g. Reinforcement Learning) represents a new powerful tool for in-silico modelling of human physiology. Biological feedback loops occurring, for instance, in peristaltic or metachronal motion, which until now could not be accounted for in in-silico models, can be tackled by the proposed technique.

1. Introduction

Computational methods such as Computational Fluid Dynamics and, more in general, Multiphysics were initially designed for problems in physics and engineering. Nowadays, however, they also are widespread in biology and medicine, and the scientific literature contains many examples of in-silico models dedicated, for instance, to the respiratory system [1,2], the cardiovascular system [3,4] or the gastrointestinal system [5,6].

There is a fundamental difference, however, between modelling engineering systems and biological systems, especially in the case of human physiology. Normally, engineering systems are fully accessible to us; they follow the well-known laws of physics and chemistry, and can be simulated simply on the basis of these laws. Human physiology also follows these laws, but has an additional layer of difficulty due to the activity of the autonomic nervous system (ANS).

The ANS acts as active feedback controller that regulates our bodily activities ensuring their correct functioning and, in general, coordinates the feedback loop between the perception of the environment and the consequent response of the body. For example, to remove microbes and particles out of the airways, pulmonary cilia move in a synchronized fashion known as metachronal wave. The cilia can sense the environment and, accordingly, readjust their movement over time to maintain the metachronal motion. This 'readjustment' is regulated by the ANS. Another example is peristalsis that occurs in the digestive tract: muscle tissue contracts in sequence to propel a mass of food down the tract. Also peristalsis requires the ability of the ANS to sense the presence of the food and act accordingly by contracting or relaxing specific parts of the tract.

The main difficulty in modelling ANS-regulated systems is that we usually do not know the 'biological algorithm' that connects the perception of the environment with the response of the system.

In the literature, this issue is generally tackled either by *ad hoc* solutions or by ignoring the feedback altogether. For example, in [2] we used Discrete Multiphysics (DMP) to model mass transfer in the respiratory ciliated layer. The synchrony of the metachronal wave was maintained over time by means of an *ad hoc* solution that takes advantage of the discrete nature of DMP. The model accounts for a 'ghost parabola' that only interacts with the cilia (Fig. 1) and, by moving along the domain, ensures the cilia never lose their coordination. This approach, however, is a sort of numerical trick that cannot be extended to other situations. Another example concerns peristalsis in the intestine. Contraction of intestinal walls follows a complex pattern that depends on the amount of food in the intestine (i.e. pressure on the wall) and its chemical composition (e.g. bitter substances increase intestinal motility). However, the actual feedback mechanism that links these factors

E-mail address: a.alexiadis@bham.ac.uk.

https://doi.org/10.1016/j.artmed.2019.06.005
Received 25 September 2018; Received in revised form 30 May 2019; Accepted 24 June 2019

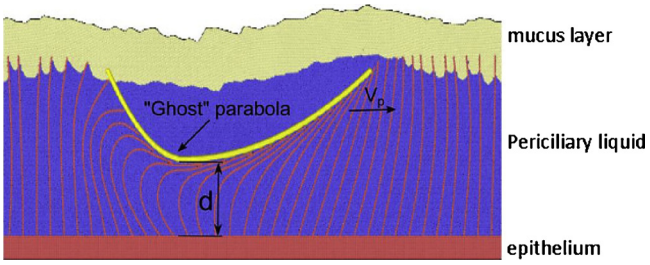

Fig. 1. The 'ghost parabola' approach used in Ariane et al. [2,18,21].

with the shape of the peristaltic wave is still unknown. To avoid this issue, [5] model transport in the small intestine only considering a completely full tract and a predetermined peristaltic wave without feedback. A DMP approach, in this case, would not help. In Alexiadis et al. [6], we proposed a DMP model of the colon capable of dealing with partially filled tracts. However, to bypass the unknown biological feedback, we validated the model with in-vitro data obtained from predetermined peristaltic waves [7].

These examples demonstrate that in-silico modelling of human physiology could greatly benefit from the ability to reverse-engineer biological feedbacks. To achieve this goal, this study proposes a methodology that combines Multiphysics and Machine Learning. The Multiphysics component of our method is based on Discrete Multiphysics (DMP), the Machine Learning component on Reinforcement Learning (RL). As benchmark/validation case, we use peristalsis in the oesophagus, which is one of the few feedback loops in human physiology that is well understood and can be modelled also without Machine Learning. To assess its validity, the model obtained by the Machine Learning algorithm is compared with a model devised without the use of Machine Learning.

Both the first principles model (discrete multiphysics) and the Artificial Intelligence algorithm (reinforcement learning) used in the benchmark are relatively well known and well tested. The real novelty of this study is, as discussed in the conclusions, the *in parallel* coupling of the two methods. From this point of view, the proposed methodology is novel at both the methodological and the theoretical level.

From the methodological point of view, its significance relates to the development of the so-called *virtual human physiology* (VHP): an interdisciplinary framework aimed at integrating computer models of the mechanical, physical and biochemical functions of a living human body. The purpose of VHP is to achieve a complete in-silico framework that will eventually lead to (i) personalized care solutions [8], (ii) prediction of chemical toxicity (that could reduce animal texting) [9], (iii) improving our understanding of treatments and diseases [10] and, in general, (iv) a more holistic approaches to medicine [11]. Within this approach, several studies coupling in-silico modelling and Machine Learning have been already published (e.g. [12,13]), but one of its main limitations remains the difficulty of accounting for the action of the autonomic nervous system. The DMP + RL approach proposed in this study can be applied to the VHP framework providing novel capabilities such as the ability of self-learning biological feedbacks and replicate their effect within in-silico models.

From the theoretical point of view, its significance relates to the development of the more general framework of *Deep Multiphysics*: a new computational framework that, based on the concept of *particle-neuron duality*, has Discrete Multiphysics and Artificial Neural Networks as special cases. This study applies to the specific instance of human physiology; for a general introduction to Deep Multiphysics, the reader is referred to Alexiadis [14].

2. Discrete Multiphysics (DMP)

2.1. A brief overview of Discrete Multiphysics

Multiphysics simulations are nowadays widespread in both industry and academia and allow tackling problems involving multiple physical models or simultaneous physical phenomena. Several commercial and open source codes (e.g. COMSOL, Elmer, Ansys Multiphysics) are available for solving multiphysics problems which, typically, involve solving coupled systems of partial differential equations on a computational mesh.

Differently from traditional multiphysics, *Discrete Multiphysics* is a mesh-free Multiphysics technique based on 'computational particles' rather than on computational meshes [15]. DMP is a hybrid approach that combines different particle methods such as Smoothed Particle Hydrodynamics (SPH), Lattice-Spring Model (MSM) and the Discrete Element Method (DEM). These techniques, in fact, follow the same computational paradigm and are particularly effective when coupled together in multiphysics applications. Typical problems where DMP has been used, involve solid-liquid flows [16] and fluid-structure interactions applications [17].

Discrete Multiphysics, however, is more than an alternative to traditional Multiphysics, and there is a variety of situations where DMP can tackle problems that would be very difficult, if not impossible for traditional multiphysics. In traditional multiphysics, sub domains are assigned during pre-processing: the user establishes before the simulation which part of the domain belongs to the solid domain and which part to the fluid domain; this choice cannot change during the simulation. In DMP, the distinction between solid and fluid depends on the type of force applied on the computational particle rather than its position on the mesh and, by changing the type of force, we can change the behavior of the particles from solid to liquid, or vice versa, *during* the simulation. This confers an advantage to Discrete Multiphysics over traditional multiphysics in cases such as modelling of cardiovascular valves including blood clotting [17,18], phase transitions [19], capsules breakup [20] and fuzzy boundaries [15].

In this paper, we show another advantage of DMP, which is its effective coupling with Machine Learning algorithms and, in particular, Reinforcement Learning.

2.2. The DMP oesophagus model

In this section, we describe the DMP oesophagus model that will be coupled with the Reinforcement Learning algorithm. The biological system comprises of a flexible tube (the oesophagus) and a central mass (the bolus). During peristalsis, the oesophagus senses the presence of the bolus and contract its smooth muscles (Fig. 2). The biological feedback coordinates these two activities with the goal of propelling the bolus down the oesophagus.

The DMP model employs the Lattice Spring Model (LSM) for the oesophagus and the discrete Element Model (DEM) for the interaction between the tube and the bolus. The wall (membrane) of the flexible tube is divided in 480 computational particles of mass m. Each of these computation particles is linked to an equilibrium point in space by a tethered spring (Fig. 3a). The equilibrium positions of the tethered springs are arranged cylindrically to represent the elastic tube at zero-

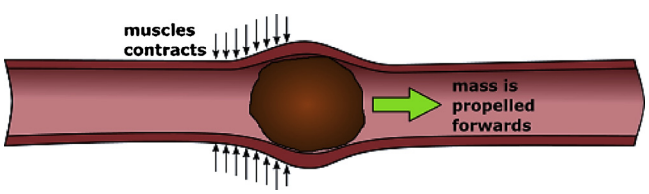

Fig. 2. Oesophagus peristalsis used as benchmark case.

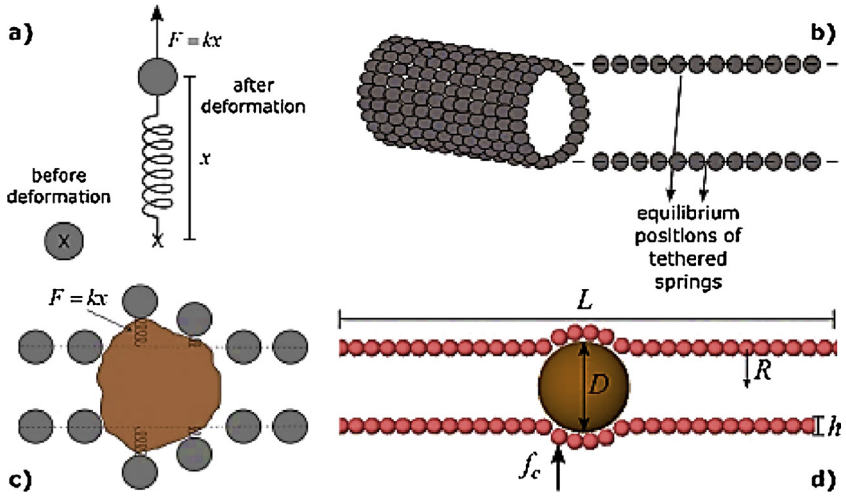

Fig. 3. Illustration of a tethered spring (a), equilibrium positions of the tethered for the flexible tube (b), deformation of the tube due to the presence of a large internal object (c), two-dimensional section of the geometry used in the simulations (d).

stress conditions (Fig. 3b). The 480 particles of the tube are divided in 20 'slices' or 'rings' with 24 particles each. When a tube-particle is displaced from its equilibrium position, the spring reacts with a Hookean force

$$F = -k_s x \tag{1}$$

that opposes the displacement x and is proportional to the spring stiffness k_s. If the flexible tube contains an object larger than its diameter, it deforms and reacts with a radial elastic force F (Fig. 3c) pointing towards the axis of the tube. The oesophagus contractions are modelled by adding radial forces f_c acting on the membrane particles. These forces are in the radial direction and point towards the axis of the tube. In Fig. 1c, f_c is applied to a single particle, but, in the 3D model, it is equally distributed to all particles belonging to the same 'slice' of the tube at a given axial position. In the simulations, the internal particle (bolus) is spherical with diameter D and mass M, the length of the tube is L, its radius R and the thickness of the membrane h (Fig. 3d).

The tube computational particles interact with the central particle by means of Hertzian contact forces consisting of a normal contact force f_n defined as

$$f_n = \sqrt{\delta}\sqrt{R_{eff}}(k_n\delta - m_{eff}\gamma_n v_n), \tag{2}$$

where $R_{eff} = r_i\, r_j\, /\, (r_i + r_j)$, is the effective radius of the colliding particles i and j with radius r_i and r_j, k_n is the stiffness of the contact, δ the overlap between particle i and j, γ_n the viscoelastic damping coefficients and $m_{eff} = m_i\, m_j\, /\, (m_i + m_j)$ the effective mass of the colliding particles with mass m_i and m_j. The concepts of overlap δ is an abstract idea that allows the DEM to calculate the forces occurring during collision (Fig. 4). The DEM model also assumes tangential forces, which we accounted for in other occasions [21], but, for simplicity, are not considered here.

There are three main features of the biological system that we want to capture in the DMP model: (i) sensory neurons located on the membrane that feel the pressure from the bolus, (ii) smooth muscles located on the membrane that can contract specific sections of the tract, and (iii) the feedback loop that coordinates the information from the sensory neurons to the muscle contractions. In the DMP model, (i) the pressure is measured by the force (or the displacement) acting on the tethered spring, and (ii) the muscle contraction is modelled by the additional forces f_c (see Fig. 1c). The feedback loop (iii), however, is lacking and will be added by means of the machine learning algorithm discussed in the next section.

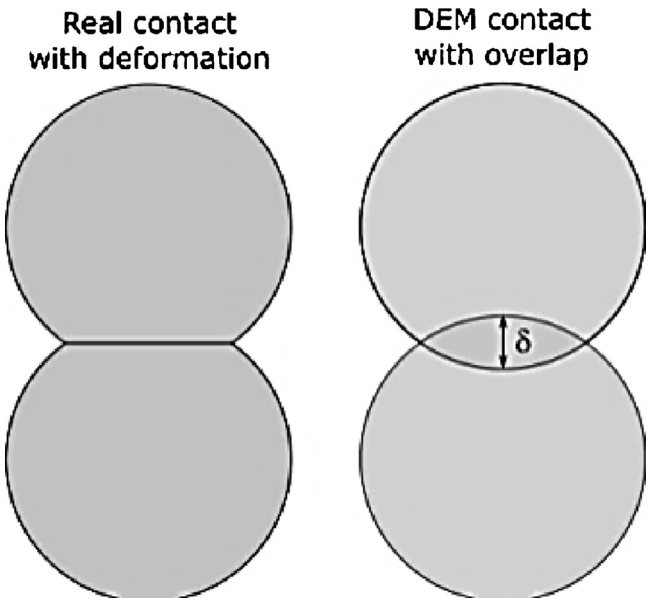

Fig. 4. Real contact with particle deformation versus DEM contact with overlap.

3. Reinforcement learning (RL)

3.1. A brief overview of Reinforcement Learning and Q-learning

Traditionally, Machine Learning is divided in three groups: Supervised Learning, Unsupervised Learning and Reinforcement Learning [22]. In Supervised Learning, we supply the machine with training data (**x**, **y**) and the model learns to map **x** to **y**. In Unsupervised Learning, we supply the machine with a set of data **x**, and the model looks for hidden patterns in the data. In Reinforcement Learning data are not necessary, the machine creates its own data by playing a sort of 'game', and the model is trained to improve its ability to play the game. This game consists of an environment, an agent, a set of states (s) that the agent can acquire, and a set of actions (a) that the agent can perform. Each action a changes the state s of the agent to a new state s'. Certain states are associated with a reward ($r > 0$), other with a penalty ($r < 0$): the goal of the agent is to find the optimal policy $\pi(a, s)$ that maps states to actions in a way that maximizes the reward while playing the game.

In this study, we use a specific Reinforcement Learning algorithm called Q-learning [23], where the policy is determined by the so-called quality Q of each action. From a given state, the agent chooses the action with the highest quality. In matrix notation, this is often represented with the so-called Q-matrix

$$Q(s, a) = \begin{bmatrix} Q(s_0, a_0) & Q(s_0, a_1) & \cdots \\ Q(s_1, a_0) & Q(s_1, a_1) & \cdots \\ \vdots & \vdots & \end{bmatrix} = Q(s_i, a_j) = Q_{i,j}. \tag{3}$$

Each element Q_{ij} indicates the quality of performing the action a_j if the agent is in state s_i. The quality is defined by the recursive algorithm (Bellman equation) that links the current state s, the action a and the new state s' reached after the action is performed

$$Q(s, a)^{new} = Q(s, a)^{old} + \alpha[r + \gamma \max_{a'} Q(s', a')^{new} - Q(s, a)^{old}]. \tag{4}$$

The term max $Q(s',a')$ returns the maximum Q value for the best possible action a' in the next state s'. In this way, the agent is looking forward to determine the best possible future rewards before deciding the current action a. The $\gamma < 1$ value is the so-called *discounting factor*; it decreases the impact of future rewards on deciding the action taken at state s. The $\alpha < 1$ value is the so-called *learning rate* and control the update of $Q(s,a)^{new}$ at each iteration.

If the Q-matrix is known, the policy that establishes the best course of action from a given state s can be easily determined from the action with maximum Q-value, e.g.

$$\pi(s, a) = \max_{a'} Q(s, a') \tag{5}$$

Initially, however, the Q-matrix is unknown and the goal of the RL algorithm is to recursively approximate the Q-matrix to a degree that allows deriving an acceptable (if not optimal) policy π. Usually, we start with an empty Q-matrix and, at first, the algorithm plays the game based on completely random choices of action. At the end of each game, it measures the reward and, by solving the Bellman equation (Eq. (4)), it gradually begins to populate the Q-matrix. If the computer would always play the game based on random actions, it would eventually explore the whole Q-matrix, but at the expenses of very long computational times. Therefore, the RL algorithm combines 'explorative' steps, based on random choices, with 'greedy' steps where the best course of action, based on the current approximation of the Q-matrix, is followed. At the beginning, with a scarcely populated Q-matrix, the RL algorithms favours exploration; as the Q-matrix gets populated, it gradually switches to a greedier approach.

3.2. Coupling the DMP model with the RL algorithm

In this section, we go from the previous, abstract, description of Q-Learning, to its concrete application within the oesophagus model. The 'game' is represented by the DMP model and the 'agent' by the bolus. The available *states*, the possible *actions*, and the consequent *rewards* are discussed hereafter.

The *state* is the position of the bolus in the tube. Apparently, we could simply assign the state as the centre of the bolus. This would be acceptable to the RL algorithm, but it would not be consistent with the biological model. In the biological model, the tube feels the presence of the bolus by perceiving the pressure against the membrane. Sensory neurons are distributed along the membrane; when the local pressure goes above a certain threshold, the neuron located at that location fires-up. The integration of the DMP with the RL algorithm should mimic this mechanism as close as possible. To achieve this, we take advantage of the discrete nature of DMP and associate a (computational) sensory neuron to each (computational) particle. In this way, each membrane's discrete element i is, at the same time, a neuron[1] and a DMP particle, which has a 'state' property s_i besides the usual DMP properties (mass m_i, velocity v_i, position r_i etc.). In Fig. 5, each neuron is associated to a particle, but, since the actual model is 3D, it represents a full 'slice' of the tube (i.e. an axisymmetric ring of particles as shown in Fig. 3b).

During the calculation, the displacement Δx_i (proportional to the elastic force F_i acting on the i^{th} particle) is measured at every time step. If $\Delta x_i > \Delta x_{MAX}$ (a threshold value) the i^{th} neuron fires-up and its state is $s_i = 1$; if $\Delta x_i < \Delta x_{MAX}$, the state is $s_i = 0$. In the RL algorithm, all the states s_i are collected in a boolean vector $\boldsymbol{s}$. To reduce the number of states, we choose Δx_{MAX} so that only one neuron (the one with maximum pressure) fires up at a time. Therefore, if only the i neuron is active, $s_i = 1$, and all the remaining elements of $\boldsymbol{s}$ are 0. This means that if the membrane is divided in N rings, the total number of different states vectors $\boldsymbol{s}$ is also N (1000 …, 0100 …, 0010 …, 0001 …, etc.).

Also for the *action*, we can take advantage of the discrete nature of DMP and associate an actions a_i with the i^{th}membrane's computational particles. In the DMP model, we model a muscle contraction at the position z_i with an additional force fc_i applied to the i^{th} computational particle (Fig. 2d). In the RL algorithm, we represent the action as a boolean vector $\boldsymbol{a}$. Each element a_i of the vector $\boldsymbol{a}$ corresponds to a membrane particle (Fig. 6): if $a_i = 1$ the i^{th} particle contracts and the force fc_i is added to the particle; if $a_i = 0$ the particle does not contract and no force is added. Differently from $\boldsymbol{s}$, where only one element at the time can be equal to 1, more than one muscle can contract at the same time and, therefore, more than one element of $\boldsymbol{a}$ can be equal to 1 at the same time. If the membrane is divided in N rings, and since each element a_i has only two possible values (1 or 0), the number of all possible actions is 2^N. For instance, if $N = 3$, the possible values of $\boldsymbol{a}$ would be (000), (100), (010), (110), (001), (101), (011), (111). Normally, N is larger than 3 (in our simulations, $N = 20$) and this is going to affect the size of the Q-matrix. We can reduce the possible values of $\boldsymbol{a}$ by noticing that only the contractions close to the bolus (i.e. where $s_i = 1$) can reasonably affect the movement of the bolus. Therefore, we can only consider a moving window of M particles around the particle with $s_i = 1$ (Fig. 6). By considering only M particles, the size of $\boldsymbol{a}$ (i.e. the number of all possible actions) reduces to 2^M. In our calculations, $M = 5$.

The *reward* is simply the shift in position Δz of the bolus after the action is executed. If the bolus advances ($\Delta z > 0$), the reward is positive; if it moves backwards ($\Delta z < 0$), the reward is negative.

Now that the *state*, the *action* and the *reward* have been defined, we can explain, with the help of Fig. 7, how the DMP model and the RL algorithm interact with each other during the simulation. At time t, the bolus is at a certain position z. The membrane's computational particles act as sensory neurons and feel the pressure due to the presence of the bolus. From the pressure, the state vector $\boldsymbol{s}$ at time t can be determined as explained above. At this point, the RL algorithm chooses the action to implement. The vector $\boldsymbol{a}$ can be determined either by the exploration or the greedy mode according to the stage of the learning process (see Section 3.1). Once the RL algorithm has chosen an action, it is implemented in the DMP model as a muscle contraction and the system evolves to the next time step. As a result of the action (i.e. muscle contraction), the bolus changes its position within the tube. The reward is calculated from the position change of the bolus Δz. One simulation runs for N_t time steps, which is not enough to achieve a good approximation of the Q-matrix. Therefore, the algorithm of Fig. 7 must be repeated many times until a good policy is obtained. Each repetition involves a new DMP simulation that runs for N_t time steps: in RL literature, a repetition is called an 'episode'.

[1] The reader should not confuse computational neurons in Fig. 5 with computational neurons typically used in Artificial Neural Networks. In this study, the term 'computational neuron' refers to the computational version of an actual neuron located on the surface of the membrane.

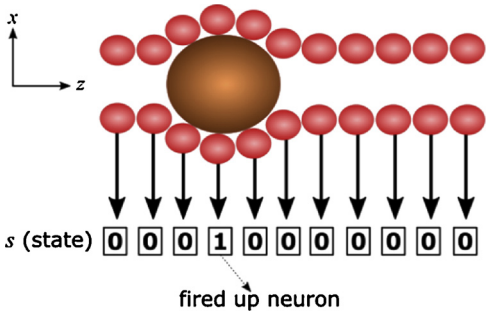

Fig. 5. Example of state vector s.

4.1. Learning phase

Training of the DMP + RL model is performed with the method discussed in Section 3.2. The goal is to move the bolus towards the positive direction of the z axis (Fig. 5), i.e. from left to the right. Fig. 8 shows the learning progress of the model with $\alpha = 0.8$ and $\gamma = 0.9$ (see Eq. (4)).

Initially, the average reward per episode is negative, which indicates a backward motion of the bolus. This occurs during the initial stage of the training, when the RL algorithm randomly explores the Q-matrix. Gradually, the algorithm learns how to propel the bolus forward and improves its performance over time. Video 1 shows how random

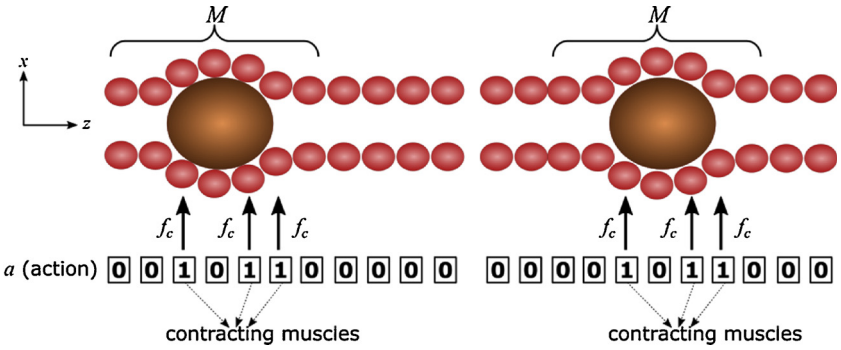

Fig. 6. Example of the (same) action vector a shifted according to the bolus position.

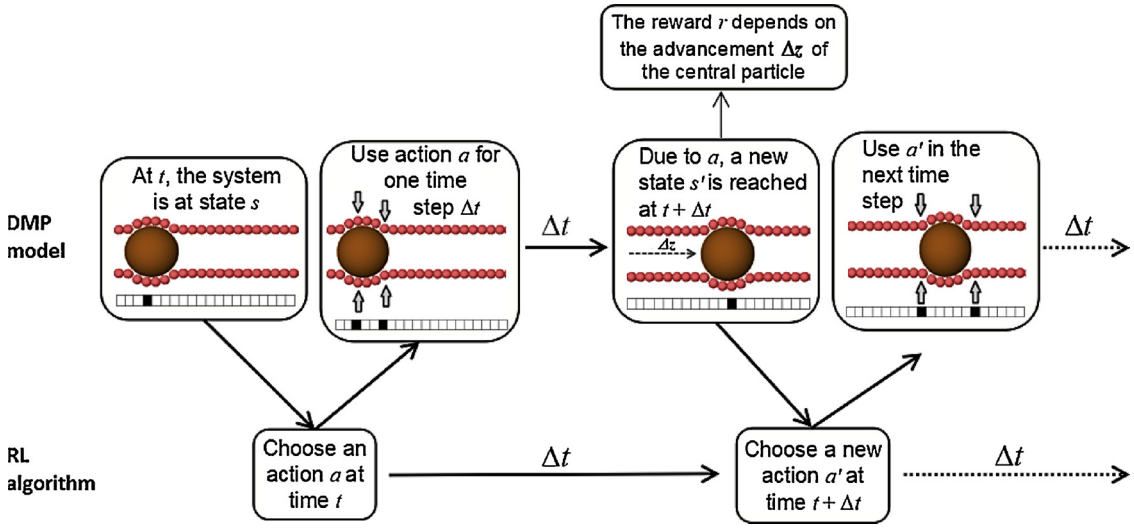

Fig. 7. How the DMP model and the RL algorithm interact during the simulation.

4. Results and discussion

In the simulations, we use (dimensionless) reduced parameters, gathered in Table 1, that refer to the case of a flexible, rubber-like, tube and a soft, gel-like, central particle. Each simulation (one episode) runs for $N_t = 400$ time steps ($\Delta t^* = 0.025$) and periodic boundary conditions are enforced in the axial direction. Gravity is not accounted for.

Since the DMP model of the oesophagus is relatively simple (compared to other DMP models), both the DMP and the RL parts of the codes are implemented in Python. This, would not be the standard approach for more complicated models. The DMP model would be implemented in software dedicated to particle simulations such as LAMMPS [24] and Python used as a wrapper to take advantage of AI libraries such as Keras.

contractions, without learning, are not coordinated and, consequently, the bolus does not move during the simulation. Video 2 shows the model at the end of the training phase, when it learnt by itself how to coordinate its contractions and propel the bolus down the oesophagus. Fig. 9 shows two screenshots of Video 2.

4.2. Man versus machine

For validation purposes, we choose peristalsis in the oesophagus because is well understood and, in this case, we can also devise a 'human' (i.e. not derived by Machine Learning) policy that links states with actions. For the sake of brevity, we call hereafter the policy obtained by the Machine Learning algorithm 'machine policy' or 'machine solution', and the policy obtained without Machine Learning 'human policy' or 'human solution'

Table 1
Reduced variables used in the simulations.

Reduced variable	Description	Value in the simulation
$m^* = m/m$	reduced mass of the membrane particles	1
$R^* = R/R$	reduced tube radius	1
$\gamma_n^* = \gamma_n/\gamma_n$	reduced viscoelastic dumping coefficient	1
$M^* = M/m$	reduced mass of the bolus	12
$t^* = t\gamma_n R$	reduced total simulation time	10
$k_s^* = \frac{k_s \gamma_n^2 R^2}{m}$	reduced stiffness of the spring (Eq. (1))	100
$k_n^* = \frac{k_n \gamma_n^2 R^3}{m}$	reduced stiffness of the contact (Eq. (2))	1000
$D^* = D/R$	reduced diameter of the bolus	2.5
$L^* = L/R$	reduced length of the flexible tube	8
$h^* = h/R$	reduced thickness of the membrane	0.4
$f_c^* = \frac{f_c \gamma_n^2 R}{m}$	reduced muscle contraction force	20

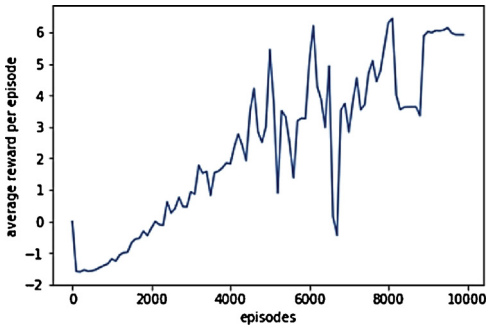

Fig. 8. Learning progress of the DMP + RL model.

The 'human policy' is obtained by contracting, in the DMP model, only the muscles behind the bolus. In this way, we 'squeeze' the bolus forwards and prevent it from moving backwards. The size of the moving window (Fig. 6) in our calculations is $M = 5$; this means that, once the particle with $s_i = 1$ has been identified, we only contract particles i-1 and i-2. In other words, in the 'human policy', the $\boldsymbol{a}$ vector remains constant and its value is $\boldsymbol{a} =$ [11000]. The contraction of the i^{th} particle (i.e. the central particle located just below the bolus) would give the highest push to the bolus. However, due to small misalignments between the bolus and the central particle, the push could move the bolus in either directions. The model does not know the exact position of the bolus z, but only its approximation z_i (the position of the neuron with $s_i = 1$). If $z < z_i$, the contraction would push the bolus on the left; if $z > z_i$, the contraction would push the bolus on the right. Since z is not known, we cannot anticipate the effect of contracting the central particle beforehand and, for this reason, the 'human policy' does not account for the contraction of the central particle.

All these considerations are not known to the machine before the learning process; and the idea of the benchmark is to verify that the machine rediscovers the correct strategy by itself and without human intervention. After comparing the policy derived by the Machine Learning algorithm with the 'human policy' (e.g. derived without Machine Learning as described above), we observed not only that the machine learns by itself how to effectively propagate the bolus, but also that the 'machine solution' is slightly better than the 'human solution', and, at the end of the simulation, the bolus advanced around 5% further with the 'machine policy' than with the 'human policy'.

How the 'machine solution' can be more efficient than the 'human solution'? Machine Learning models are black-box models and usually it is not easy to understand the actual physics behind their functioning. In our DMP + RL model, however, computational particles and neurons are highly integrated, and this allows a deeper look inside the physical effects of the machine strategy.

At the beginning, when the bolus is at rest, both the 'machine' and the 'human' policies are the same (e.g. $\boldsymbol{a} =$ [11000]). When the bolus reaches a certain velocity, however, the 'machine solution' (contrary to the 'human solution') also contracts the central particle (e.g. $\boldsymbol{a} =$ [11100]). The machine 'understands' that, at the beginning, when the bolus is at rest, contracting the central particle can be counterproductive. However, it improves the 'human solution' by noticing that, if the bolus is already moving in the desired direction and its velocity is above a certain value, contracting the central particle becomes beneficial. In fact, there is a delay between when the contraction begins and when the bolus fells the squeeze. Even if $z < z_i$ at the beginning of the contraction, the bolus moves forward and, at the moment of the squeeze, its location is $z > z_i$. In this way, the contraction pushes the bolus forward even if initially $z < z_i$.

5. Conclusions

This study proposes a methodology for designing in-silico models of human physiology with the ability of learning by themselves how to reproduce biological feedbacks. The proposed methodology combines Discrete Multiphysics and Reinforcement Learning and, to the best of our knowledge, it is the first time multyphisics and machine learning are *coupled in parallel* rather than *in series* (Fig. 10).

Previous works (e.g. [25–30]) combine first-principles models (e.g. CFD or traditional multiphysics) with Machine Learning algorithms (e.g. Artificial Neural Networks) in series. The fist-principle model provides data that are used, instead of experimental or real-world data, for training the Artificial Neural Networks (ANN). Once the ANN is

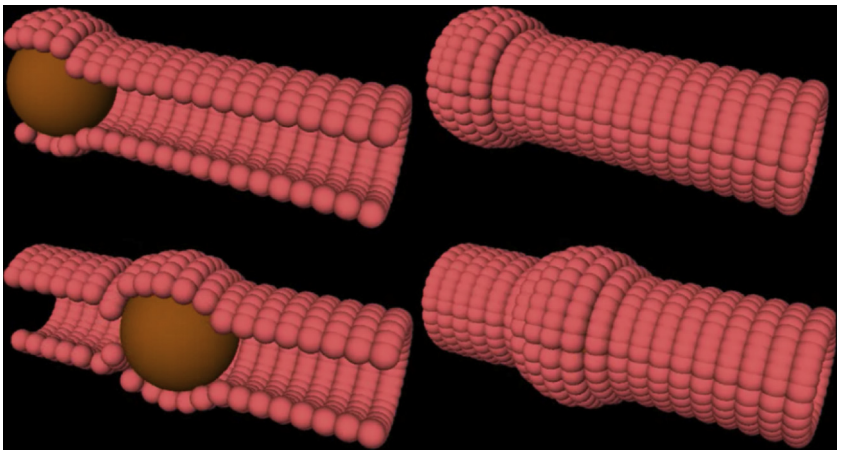

Fig. 9. DMP model after RL training: at $t^* = 0$ (top) and $t^* = 2.5$ (bottom); part of the wall is removed from the pictures on the left to show the position of the bolus.

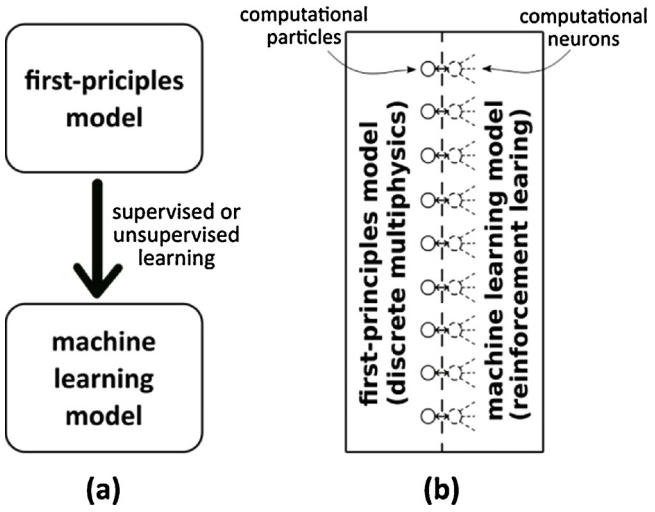

Fig. 10. *In series* coupling (a) versus *in parallel* coupling (b) of first-principles and machine learning models.

trained, it can replace the first-principle model and, ideally, provide a model that is computationally faster than the first-principle one. Although we could not find any example in the literature (all the articles above are based on supervised learning), an alternative in series approach could be based on unsupervised learning. The data coming from the first-principle model could be used for reducing the dimensionality of a physical problem (e.g. turbulence in a specified system) and, in this way, reduce the number of actual variables required to define the problem.

In both these scenarios, the two models remain conceptually separated. The data of the first-principle model are transferred to the machine learning model after one or more simulations have been completed and, for this reason, the coupling is in series.

In this study, the coupling between the two models occurs at a deeper level. The computational particles used in Discrete Multiphysics to model the biological system represent, at the same time, computational neurons. This circumstance allows Reinforcement Learning to train the neurons as the simulation progresses (see Fig. 7) and, for this reason, the coupling is defined in parallel. In other words, while *in series coupling produces two models* of the same system (a first-principles and a machine learning model) with the goal of providing a faster computational tool (or alternatively for control/optimization purposes), *in parallel coupling produces a single hybrid model with two components* (a first-principles and a machine learning component) with the goal of providing a model capable of solving complex problems that could not be tackled with first-principles modelling or machine learning separately.

The benefits of in parallel coupling can be observed from two different perspectives. On the one hand, Machine Learning can help first-principle models designing better in-silico models of human physiology by providing a way of replicating the biological feedback loops regulated by the autonomic nervous system. On the other hand, first-principle models can help machine learning to achieve more 'transparent' models. Machine Learning models, in fact, are black-box models; they do work, but, since there is not physics or chemistry behind, we usually do not have a clear understanding on why they works. The particle/neuron duality, however, provides an unprecedented access to the physical effects of the computational neurons and, as exemplified in Section 4.2, this allows to understand why and how the strategy proposed by the machine learning algorithm actually works. Moreover, here we focuse on a benchmark case (peristalsis in the oesophagus), but future work will extend to more complex systems. In these cases, Reinforcement Learning will probably be replaced by Deep Reinforcement Learning and the duality computational-particle/computational-neuron will become even more significant.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant number: EP/S019227/1.

Appendix A. Supplementary data

Supplementary material related to this article can be found, in the online version, at doi:https://doi.org/10.1016/j.artmed.2019.06.005.

References

[1] Koullapis PG, Nicolaou L, Kassinos SC. In silico assessment of mouth-throat effects on regional deposition in the upper tracheobronchial airways. J Aerosol Sci 2018;117:164–88.
[2] Ariane M, Kassinos SC, Velaga S, Alexiadis A. Discrete multi-physics simulations of diffusive and convective mass transfer in boundary layers containing motile cilia in lungs. Comput Biol Med 2018;95:34–42.
[3] Quarteroni A, Manzoni A, Vergara C. The cardiovascular system: mathematica modeling, numerical algorithms. Clin Appl Acta Numer 2017;26:365–590.
[4] Ariane M, Allouche H, Bussone M, Giacosa F, Bernard F, Barigou M, et al. Discrete multiphysics: a mesh-free approach to model biological valves including the formation of solid aggregates at the membrane surface and in the flow. PLoS One 2017;12:e0174795.
[5] Sinnott MD, Cleary PW, Arkwright JW, Dinning PG. Investigating the relationships between peristaltic contraction and fluid transport in the human colon using Smoothed Particle Hydrodynamics. Comput Biol Med 2012;42:492–503.
[6] Alexiadis A, Stamatopoulos K, Wen W, Bakalis S, Barigou M, Simmons M. Using discrete multi-physics for detailed exploration of hydrodynamics in an in vitro colon system. Comput Biol Med 2017;81:188–98.
[7] Stamatopoulos K, Batchelor HK, Simmons MJH. Dissolution profile of theophylline modified release tablets, using a biorelevant Dynamic Colon Model (DCM). Eur J Pharm Biopharm 2016;108:9–17.
[8] Sadiq SK, Mazzeo MD, Zasada SJ, Manos S, Stoica I, Gale CV, et al. Patient-specific simulation as a basis for clinical decision-making. Philos Trans R Soc A 2008;366:3199–219.
[9] Raies AB, Bajic VB. In silico toxicology: computational methods for the prediction of chemical toxicity. Wiley Interdiscip Rev Comput Mol Sci 2016;6:147–72.
[10] Jean-Quartier C, Jeanquartier F, Jurisica I, Holzinger A. In silico cancer research towards 3R. BMC Cancer 2018;18:408–20.
[11] Clapworthy G, Viceconti M, Coveney PV, Kohl P. The virtual physiological human: building a framework for computational biomedicine. Philos Trans R Soc A 2008;366:2975–8.
[12] Fox T, Kriegl JM. Machine learning techniques for in silico modeling of drug metabolism. Curr Top Med Chem 2006;6:1579–91.
[13] Jeanquartier F, Jean-Quartier C, Kotlyar M, Tokar T, Hauschild AC, Jurisica I. Machine learning for in silico modeling of tumor growth. Mach Learn Health Inf 2016;9605:415–34.
[14] Alexiadis A. From Discrete Multiphysics to Deep Multiphysics: a case study concerning the design of continuous microfluidic devices for cell separation. Appl. Math. Modell. 2019 submitted (preprint available at https://hal.archives-ouvertes.fr/hal-02143375.
[15] Alexiadis A. A smoothed particle hydrodynamics and coarse-grained molecular dynamics hybrid technique for modelling elastic particles and breakable capsules under various flow conditions. Int J Numer Methods Eng 2014;100:713–9.
[16] Alexiadis A. The Discrete Multi-Hybrid System for the simulation of solid-liquid flows. PLoS One 2015;10:e0124678.
[17] Ariane M, Wen W, Vigolo D, Brill A, Nash GB, Barigou M, et al. Modelling and simulation of flow and agglomeration in deep veins valves using Discrete Multi Physics. Comput Biol Med 2017;89(1):96–103. 2015.
[18] Ariane M, Vigolo D, Brill A, Nash GB, Barigou M, Alexiadis A. Using Discrete Multi-Physics for studying the dynamics of emboli in flexible venous valves. Comput Fluids 2018;166:57–63.
[19] Alexiadis A, Ghraybeh S, Geng Q. Natural convection and solidification of phase-change materials in circular pipes: a SPH approach. Comput Mater Sci 2018;150:475–83.
[20] Alexiadis A. A new framework for modelling the dynamics and the breakage of capsules, vesicles and cells in fluid flow. Procedia UTAM 2015;16:80–8.
[21] Ariane M, Sommerfeld M, Alexiadis A. Wall collision and drug-carrier detachment in dry powder inhalers: using DEM to devise a sub-scale model for CFD calculations. Powder Technol 2018;333:65–75.
[22] Raschka S, Mirjalili V. Python machine learning. 2nd edition Packt Publishing; 2017.
[23] Akhtar FSM. Practical reinforcement learning. Packt Publishing; 2017.
[24] Plimpton S. Fast parallel algorithms for short-range molecular dynamics. J Comput Phys 1995;117(1995):1–19.
[25] Yi YK, Malkawi AM. Integrating neural network models with computational fluid dynamics (CFD) for site-specific wind condition. Build Simul 2011;4:245–53.
[26] Gholami A, Bonakdari H, Zaji H, Akhtari AA. Simulation of open channel bend characteristics using computational fluid dynamics and artificial neural networks. Eng Appl Comput Fluid Mech 2015;9:355–69.
[27] Zhang ZJ, Duraisamy K. Machine learning methods for data-driven turbulence

modeling. 22nd AIAA computational fluid dynamics conference 2015.
[28] Mason L, Pan I, Karnik A, Batchvarov A, Craster R, Matar O. Will it flood? Classifying entrainment outcomes via machine learning. Bull Am Phys Soc 2018 available on-line at http://meetings.aps.org/Meeting/DFD18/Session/G32.1.
[29] Villasana M, Ochoa G, Aguilar S. Modeling and optimization of combined cytostatic and cytotoxic cancer chemotherapy. Artif Intell Med 2010;50:163–73.
[30] Tonutti M, Gras G, Yang GZ. A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery. Artif Intell Med 2017;80:39–47.