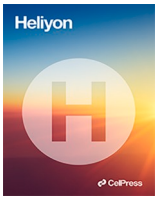
Heliyon 10 (2024) e28822

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Research article

# Preserving multi-dimensional information: A hypersphere method for parameter space analysis

Nicolas A.C. Davey *, J. Geoffrey Chase, Cong Zhou, Liam Murphy

*University of Canterbury, New Zealand*

ARTICLE INFO

*Keywords:*
Cardiovascular
Hypersphere
Dimensionality
Parameter space analysis
Visualisation

ABSTRACT

*Background:* Physiological modelling often involves models described by large numbers of variables and significant volumes of clinical data. Mathematical interpretation of such models frequently necessitates analysing data points in high-dimensional spaces. Existing algorithms for analysing high-dimensional points either lose important dimensionality or do not describe the full position of points. Hence, there is a need for an algorithm which preserves this information.
*Methods:* The most-distant uncovered point (MDUP) hypersphere method is a binary classification approach which defines a collection of equidistant N-dimensional points as the union of hyperspheres. The method iteratively generates hyperspheres at the most distant point in the interest region not yet contained within any hypersphere, until the entire region of interest is defined by the union of all generated hyperspheres. This method is tested on a 7-dimensional space with up to 35.8 million points representing feasible and infeasible spaces of model parameters for a clinically validated cardiovascular system model.
*Results:* For different numbers of input points, the MDUP hypersphere method tends to generate large spheres away from the boundary of feasible and infeasible points, but generates the greatest number of relatively much smaller spheres around the boundary of the region of interest to fill this space. Runtime scales quadratically, in part because the current MDUP implementation is not parallelised.
*Conclusions:* The MDUP hypersphere method can define points in a space of any dimension using only a collection of centre points and associated radii, making the results easily interpretable. It can identify large continuous regions, and in many cases capture the general structure of a region in only a relative few hyperspheres. The MDUP method also shows promise for initialising optimisation algorithm starting conditions within pre-defined feasible regions of model parameter spaces, which could improve model identifiability and the quality of optimisation results.

## 1. Introduction

Physiological modelling often creates models described by relatively large numbers of variables and/or data from various clinical measurements. Frequently it is advantageous to represent this data as points in a high-dimensional space, to enable efficient mathematical analysis. However, such high-dimensional points are difficult to intuitively understand or visualise when straightforward plotting exceeds three dimensions.

* Corresponding author.
*E-mail address:* nicolas.davey@pg.canterbury.ac.nz (N.A.C. Davey).

https://doi.org/10.1016/j.heliyon.2024.e28822
Received 15 March 2024; Accepted 25 March 2024
Available online 30 March 2024

Dimension reduction methods, such as principal component analysis (PCA) [1] or T-Distributed Stochastic Neighbour Embedding (t-SNE) [2], can ameliorate the issue, but can struggle with nonlinear system dynamics and may reduce dimensionality along combinations of variables which are not physiologically meaningful. Similarly, when large datasets are considered, clustering algorithms, such as k-means clustering [1], can show where many points lie or how they are grouped, but do not eliminate the issue of dimensionality for intuitively understanding the layout of points. Parallel coordinate plots [3,4] can visualise high-dimensional data, but distort the view of the parameter space and can quickly become cluttered with multiple data points. Both of these characteristics make intuitive interpretation of lager patterns within the parameter space a significant challenge with such plots. Thus, a reliable method for extracting useful and intuitive information about the distribution of large numbers of high-dimensional points is needed to interpret patterns and correlations in such datasets, and has been identified as a gap in parameter space analysis research [5].

Several methods have been developed with this aim in mind. Glyph-based visualisations [6] can extend into higher dimensions, but do not scale reliably when numerous additional dimensions require visualisation. Evers and Linsen [7] propose a method of visualising high-dimensional parameters with interactive hyper-slicer visualisations. While this presents several advantages over existing methods, users can still miss key information if the parameter space does not neatly align with the 'slices' being used to visualise it. The hyper-slicer method is also noted to struggle with higher-dimensional parameter spaces due to computational load. Region-growing approaches [8] provide a means of defining a space in its full dimensionality, but due to their potential geometric complexity struggle to add intuition to the layout of a parameter space without further processing.

The three-chamber model (3CM) is a cardiovascular model representing the cardiovascular system as three elastic chambers connected via flow resistances [9–12]. Due to its highly simplified construction compared to the full scope of the cardiovascular system and other cardiovascular models [13–17], the 3CM can simulate the major dynamics of cardiovascular pressures and flows using only 7 input parameters, allowing it to describe cardiovascular states with low computational load. However, 7 variables remains beyond easy visualisation, and there is some parameter trade-off [18], even if the 3CM is one of the simplest physiologically relevant cardiovascular system models [19].

Identifying models with parameter trade-off and relatively limited data requires regularisation to reliably implement optimisation algorithms such as gradient descent or Newton's method. Newer approaches can use machine learning methods to identify complex models [20–22]. However, clinical human data can be relatively scarce compared to the potential number of variable value combinations, meaning training such machine learning algorithms can be difficult or may be incomplete. In addition, accurate labelling of human data can be unrealistic in large volumes.

One method to avoid this issue is to use the model itself to generate large amounts of synthetic data with automated ground-truth labelling. The synthetic data can be used to train machine learning methods, such as support vector machines, before refining them with scarcer human data. This approach has been successfully employed in nonlinear structural health monitoring applications and could be relevant to similarly nonlinear physiological models [20]. Using the model to generate synthetic training data requires extensive offline simulation, but does not impact efficiency once training is complete. To ensure accuracy in this approach, a reliable method for analysing the distribution and structure of synthetic data is essential to refine and optimise its use.

In particular, synthetic data based on random selection or chosen from a grid of parameter value combinations may produce model results which are not feasible or realistic. Thus, there would arise synthetic data creating feasible and infeasible regions of model parameter combinations, which in themselves shed insight into model structure and function. As noted, numerous methods exist for analysing the distribution of points in high-dimensional space, but no existing method provides insight into the groupings of points in their full high-dimensional state.

This research explores a novel method for extracting useful information about the groupings of high-dimensional points, in the

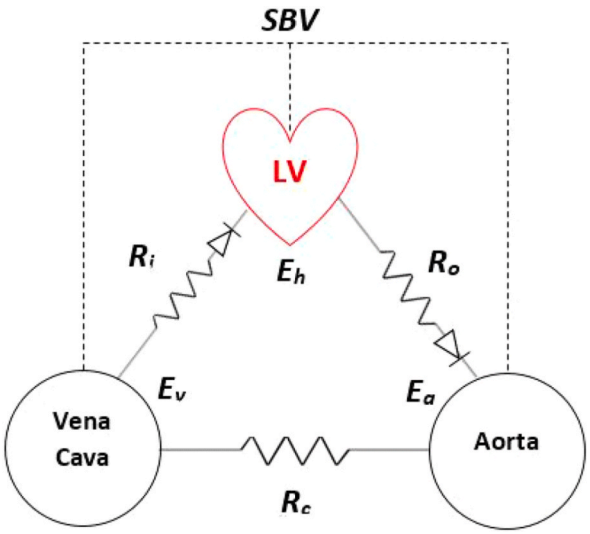



**Fig. 1.** The three-chamber cardiovascular model. Figure taken from Ref. [10].

context of analysing physiologically feasible and infeasible model parameter inputs to the 3CM. The goal is to define the feasible and infeasible parameter value spaces to provide optimal starting point selections for standard optimisation-based system identification of the model, and to potentially aid machine learning based identification using support vector machines (or similar).

## 2. Methods

Simulating the 3CM with different input combinations always yields an output. However, it is necessary to distinguish between inputs resulting in physiologically feasible outputs, and those which do not. This differentiation creates feasible and infeasible 7-dimensional (7D) parameter spaces.

### 2.1. Model description

The 3CM (Fig. 1) describes the circulatory system as a series of elastic chambers representing the heart (left ventricle/LV), arteries (aorta), and veins (vena cava). Pressure in the arterial and venous chambers are described using a linear relationship between the stressed blood volume in the chamber and a constant chamber elastance (E).

$$P_a(t) = E_a * SBV_a(t) \tag{1}$$

$$P_v(t) = E_v * SBV_v(t) \tag{2}$$

An additional time-varying elastance term is used to describe the cardiac chamber, to account for the heart's ability to spontaneously contract and relax.

$$P_h(t) = e(t) * E_h * SBV_h(t) \tag{3}$$

Blood flow from one chamber to the next is described using an Ohm's Law relationship between the pressure in the respective chambers to describe the resistance (R) to flow between them. Flow into, and out of, the cardiac chamber is restricted by cardiac valves represented in the 3CM as two ideal diodes, shown in Fig. 1 and represented by the Heaviside function $H$

$$Q_c(t) = \frac{P_a(t) - P_v(t)}{R_c} \tag{4}$$

$$Q_i(t) = H(P_v(t) - P_h(t)) * \frac{P_v(t) - P_h(t)}{R_i} \tag{5}$$

$$Q_o(t) = H(P_h(t) - P_a(t)) * \frac{P_h(t) - P_a(t)}{R_o} \tag{6}$$

Finally, the total stressed blood volume (SBV) in the circulation is assumed to remain constant throughout on cardiac cycle, with blood either flowing from one chamber to the next or accumulating in a chamber causing an associated increase in pressure [9–12].

$$SBV_h(t) + SBV_a(t) + SBV_v(t) = SBV \tag{7}$$

Thus, given the parameter set $p = \{E_a, E_v, E_h, R_i, R_o, R_c, SBV\}$ the 3CM can be used to simulate pressure and volume throughout the systemic circulation.

Detailed model equations can be found in references [9,10,12,18,23].

### 2.2. Data generation and pre-processing

To simulate a full range of cardiovascular states, a set of potential values for each 3CM input parameter was determined between a physiologically informed maximum value based on prior experience in identifying values from clinical data [12,23–25], and a minimum value of zero. Each parameter was also given a step size, to determine the granularity of analysis. To avoid physiologically meaningless input values of zero, the step size was used as the minimum value for each parameter. The stepped 7 input parameter values represent points in a 7D grid.

**Table 1**
Maximum values used for each parameter, with example step size and minimum values.

| Parameter | Maximum | Step Size | Minimum |
|---|---|---|---|
| $E_h$ (mmHg ml$^{-1}$) | 3 | *0.5* | *0.5* |
| $E_a$ (mmHg ml$^{-1}$) | 3 | *0.5* | *0.5* |
| $E_v$ (mmHg ml$^{-1}$) | 0.5 | *0.1* | *0.1* |
| $R_i$ (mmHg s ml$^{-1}$) | 1 | *0.1* | *0.1* |
| $R_o$ (mmHg s ml$^{-1}$) | 0.1 | *0.1* | *0.1* |
| $R_c$ (mmHg s ml$^{-1}$) | 2 | *0.1* | *0.1* |
| SBV (ml) | 200 | *50* | *50* |

With all potential parameter values established, each combination is input into the 3CM and run for 25 cardiac cycles to allow the model to reach steady state. The model is run using a cardiac driver function, describing the time-varying elastance of the cardiac chamber, derived from pressure and volume data obtained from porcine experimental data [10]. After simulation, each combination of parameter values yields a set of arterial and venous pressure waveforms, which are assessed as either clinically feasible or infeasible. An example set of parameters used in this analysis are shown in Table 1, and a complete list of step sizes alongside the total number of generated points is included in Table 2. The feasibility criteria are shown in Table 3 and were created in consultation with ICU Senior Specialists at Christchurch Hospital, Christchurch, New Zealand.

In Table 3, arterial pressure ($P_a$) refers to the pressure in the Aorta chamber, and likewise venous pressure ($P_v$) refers to the pressure in the Vena Cava chamber. Pulse pressure ($PP$) is thus defined:

$$PP = \max(P_a) - \min(P_a) \tag{8}$$

And contractility ($C$) is defined:

$$C = \max(\nabla P_a) \tag{9}$$

Finally, the full set of 3CM equations for simulation are found in Refs. [9,10].

Once each input point was labelled as feasible or infeasible using the criteria in Table 3, the grid of input points was scaled by the step size along each axis, so the distance between points in all seven axes was 1. This scaling ensured equal weight for all parameters in distance calculations used in the algorithm presented in the next section, and enabled the input parameter values to be stored as unsigned integers to make efficient use of memory.

### 2.3. Most-distant uncovered point (MDUP) hypersphere method

The hypersphere method defines a set of high-dimensional points as the union of the volume contained by a collection of hyperspheres. Because a hypersphere can be defined purely as a centre and a radius, its properties can be easily understood in a space of any dimension. Therefore, defining the feasible and infeasible spaces as a collection of hyperspheres can grant insights into the structure of the space other methods do not provide.

The Most-Distant Uncovered Point (MDUP) hypersphere method simultaneously fills the region of interest with hyperspheres and monitors which points are 'covered' and 'uncovered' by the union of the volumes of all created hyperspheres. Following discussion will focus on filling the infeasible space for demonstration purposes, but the method is identical when filling the feasible space (also note that in this example the feasible and infeasible space sets are complementary, so defining one also defines the other). The method is defined by the following steps.

1. Select a corner point in the infeasible set. This point will be the centre of the first hypersphere.
2. Calculate the Euclidean distance to the nearest point in the feasible set. This defines the radius of the hypersphere centred on the selected point. The equation to calculate the Euclidean distance between two points is defined:

**Table 2**
Step sizes and resulting total number of generated points. Note that the number of feasible points and the number of infeasible points sum to the total number of generated points.

| Step Size | | | | | | | Generated Points | |
|---|---|---|---|---|---|---|---|---|
| $E_h$ | $E_a$ | $E_v$ | $R_i$ | $R_o$ | $R_c$ | SBV | Total Points (P) | Total Feasible Points |
| $\frac{3}{5}$ | $\frac{3}{5}$ | $\frac{0.5}{5}$ | $\frac{1}{6}$ | $\frac{0.1}{6}$ | $\frac{2}{6}$ | $\frac{200}{6}$ | 162,000 | 6133 |
| $\frac{3}{6}$ | $\frac{3}{6}$ | $\frac{0.5}{6}$ | $\frac{1}{6}$ | $\frac{0.1}{6}$ | $\frac{2}{6}$ | $\frac{200}{6}$ | 279,936 | 9620 |
| $\frac{3}{7}$ | $\frac{3}{7}$ | $\frac{0.5}{7}$ | $\frac{1}{7}$ | $\frac{0.1}{7}$ | $\frac{2}{7}$ | $\frac{200}{7}$ | 823,543 | 28,811 |
| $\frac{3}{7}$ | $\frac{3}{7}$ | $\frac{0.5}{7}$ | $\frac{1}{8}$ | $\frac{0.1}{8}$ | $\frac{2}{8}$ | $\frac{200}{8}$ | 1,404,928 | 51,514 |
| $\frac{3}{8}$ | $\frac{3}{8}$ | $\frac{0.5}{8}$ | $\frac{1}{8}$ | $\frac{0.1}{8}$ | $\frac{2}{8}$ | $\frac{200}{8}$ | 2,097,152 | 73,936 |
| $\frac{3}{8}$ | $\frac{3}{8}$ | $\frac{0.5}{8}$ | $\frac{1}{9}$ | $\frac{0.1}{9}$ | $\frac{2}{9}$ | $\frac{200}{9}$ | 3,359,232 | 122,098 |
| $\frac{3}{9}$ | $\frac{3}{9}$ | $\frac{0.5}{9}$ | $\frac{1}{9}$ | $\frac{0.1}{9}$ | $\frac{2}{9}$ | $\frac{200}{9}$ | 4,782,969 | 168,259 |
| $\frac{3}{9}$ | $\frac{3}{9}$ | $\frac{0.5}{9}$ | $\frac{1}{10}$ | $\frac{0.1}{10}$ | $\frac{2}{10}$ | $\frac{200}{10}$ | 7,290,000 | 262,507 |
| $\frac{3}{9}$ | $\frac{3}{9}$ | $\frac{0.5}{10}$ | $\frac{1}{10}$ | $\frac{0.1}{10}$ | $\frac{2}{10}$ | $\frac{200}{10}$ | 8,100,000 | 289,853 |
| $\frac{3}{10}$ | $\frac{3}{10}$ | $\frac{0.5}{10}$ | $\frac{1}{10}$ | $\frac{0.1}{10}$ | $\frac{2}{10}$ | $\frac{200}{10}$ | 10,000,000 | 351,773 |
| $\frac{3}{11}$ | $\frac{3}{11}$ | $\frac{0.5}{11}$ | $\frac{1}{11}$ | $\frac{0.1}{11}$ | $\frac{2}{11}$ | $\frac{200}{11}$ | 19,487,171 | 687,688 |
| $\frac{3}{12}$ | $\frac{3}{12}$ | $\frac{0.5}{12}$ | $\frac{1}{12}$ | $\frac{0.1}{12}$ | $\frac{2}{12}$ | $\frac{200}{12}$ | 35,831,808 | 1,265,034 |

**Table 3**
Requirements for output to be deemed physiologically feasible.

| Parameter | Requirements for Feasibility |
|---|---|
| Maximum Arterial Pressure | <200 mmHg |
| Minimum Arterial Pressure | <20 mmHg |
| Maximum Venous Pressure | <20 mmHg |
| Maximum Pulse Pressure | <200 mmHg |
| Minimum Pulse Pressure | >20 mmHg |
| Contractility | >50 mmHg $ml^{-1}$ |

$$d = \sqrt{\sum_{i=1}^{N} (p_i - q_i)^2} \tag{10}$$

where $N$ is the number of dimensions with $N = 7$ in this case, and $p$ and $q$ are the $N$-dimensional vectors representing the two points. In this example the point $p$ represents the current centre point, and $q$ is a point in the feasible set.

The nearest point is the point $q$ in the feasible set which produces the smallest value of $d$ for the current centre point $p$. The distance $d$ to the nearest point defines the radius $r$ of the hypersphere:

$$rr(p) = min_{qinF} d(p, q) \tag{11}$$

3. Mark all infeasible (feasible) points in the volume of this hypersphere as 'covered' and remove them from the set of infeasible points remaining. A point $u$ is considered 'covered' if it satisfies:

$$\sqrt{\sum_{i=1}^{N} (p_i - u_i)^2} \leq r \tag{12}$$

given the values of $p$ and $r$ from Step 2.

4. Select from the remaining 'uncovered' infeasible points, the infeasible point with the greatest Euclidean distance from the current centre point (Eq. (3)). This point will be set as the centre point $p$ of the next hypersphere.
5. Repeat steps 2–4 until all infeasible points are marked as covered, and the full infeasible region is defined by the union of the set of created hyperspheres.

An example of this method for an easily visualised 2D case is shown in Fig. 2. This example uses two discrete parameters ranging from 1 to 12, with a step size of 1. The feasible and infeasible regions are arbitrarily defined for this example.

### 2.4. Computational experiment setup

Using data generated from the 3CM, the MDUP method was run on a local machine running Microsoft Windows 10 Enterprise with a i7-10700 CPU (2.90 GHz) and 64 GB of RAM. Code was written and executed in MATLAB 2022a.

### 2.5. Analyses

The total number of input points $P$ is given by

$$P = \prod_{i=1}^{N} \left( \frac{a_i - b_i}{s_i} \right) \tag{13}$$

where $a$ is the vector of maximum parameter values (Table 1), $b$ is the vector of minimum parameter values, $s$ is the vector of parameter step sizes, and $N$ is the number of input parameters/dimensions ($N = 7$ in this case).

Using data generated from the 3CM generates matrices with $N$ columns and $P$ rows. The value of $P$ depends entirely on the resolution of the step size used when working with fixed maximum and minimum parameter values, as shown in Table 2. The results for the infeasible and feasible regions are examined for 12 different combination of step sizes given in Table 2. Results are presented in increasing order of size, followed by a computational time sensitivity analysis. To demonstrate the algorithm, both the feasible and infeasible spaces were examined. This analysis thus shows the sensitivity analysis across the number of points with changing step sizes and the resulting size of the data set.

As the MDUP method creates hyperspheres defined by radius, Table 4 shows the number of points contained inside the volumes of 7D hyperspheres with different radii for reference.

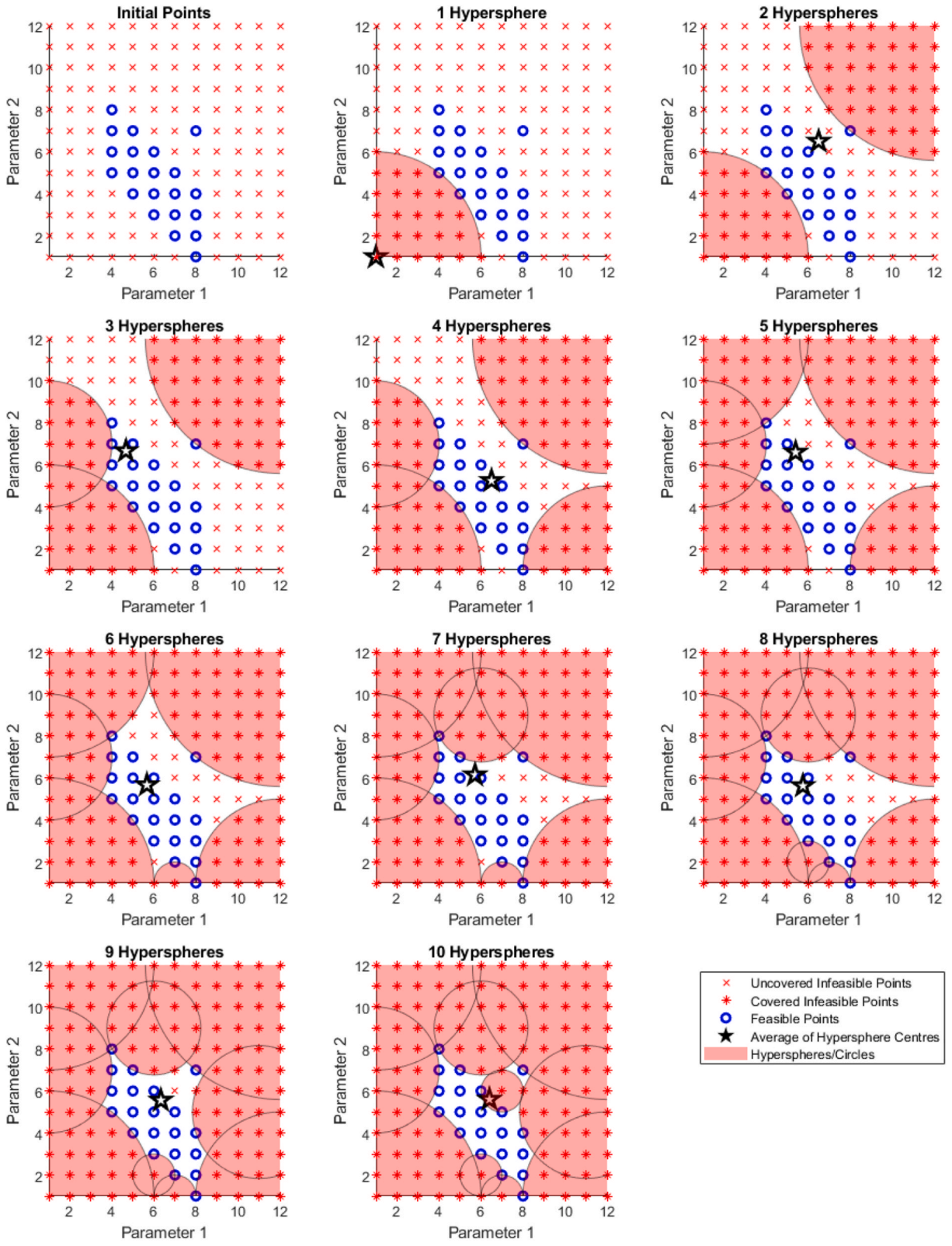



**Fig. 2.** An example of the MDUP Method in a 2D grid.

## 3. Results

The very low percentage of feasible values in Table 2 is uniquely interesting in its own right from a system identification and physiological modelling perspective, as all input parameters were within feasible ranges based on identified values found in prior research using this model [9,10,12,18,23]. It thus indicates the need for defining a feasible region well to pick a better or best starting point for any optimisation and identification problem. Further, it shows only a relatively limited number of model generated points (<3.5% in this case) might be required for any form of machine learning system to be trained to identify model parameters from measured, real data, which is important to know in terms of size and ensuring the synthetic data spans the feasible region well.

The number of hyperspheres generated and their radii are shown in Tables 5 and 6. The minimum possible radius is 1, comprising 15 combinations of parameter values per Table 4. The step sizes used to generate these values are provided in Table 2 based on the cases defined. The times taken for the algorithm to complete with different numbers of total input points are shown in Fig. 3, alongside least squares regression approximations.

The cumulative sum of the number of covered points added by each hypersphere as they were generated are shown in Figs. 4 and 5, for the infeasible and feasible space, respectively. When hyperspheres are sorted in the order they were generated, it is equivalent to the total number of points contained in the union of all generated hyperspheres (left). The cumulative sum of the number of covered points added by each hypersphere in descending order of radius is also shown right. Figs. 6 and 7 outline the number of covered points added by the 50 largest-radius spheres when defining the infeasible and feasible space, respectively. The centre points of the largest 10 spheres are presented in Tables 7–9.

## 4. Discussion

### 4.1. Hypersphere size distribution

A median and mode of the hypersphere radii of 1 suggests a majority of the infeasible region was covered by a small number of large-radius hyperspheres. Tables 5 and 6 show the median and mode hypersphere radii are consistently the minimum possible value, indicating most generated hyperspheres were likely clustered around the border of the feasible region, adding only a few points at a time. Fig. 4 shows the largest 14% of spheres cover more than 90% of all points, while Fig. 6 further reinforces the dominance of the few largest spheres, showing a rapid decrease in the number points added after only the largest 50 spheres.

However, is worth noting a point was not considered added if it was already covered by a previously generated hypersphere. In many cases, this approach reduces the apparent contribution of a hypersphere if it directly borders (and therefore overlaps) an existing hypersphere, and thus adds fewer points. For this reason, the algorithm starts each hypersphere at the farthest distance from prior created hyperspheres to try and find the biggest hyperspheres first, thus reducing the number of 7D points or parameter sets to be examined more rapidly.

Fig. 4 highlights there is no guarantee of generating hyperspheres in order of importance or size with this approach. Even after several hundred spheres had been generated, there are still instances of some spheres being generated with a relatively larger radius and adding numerous points. However, the first spheres cover a measurable majority of points, and modifying the algorithm to seek out the largest spheres first would add significant complexity for only modest gain in speed of coverage. Considering how to parallelise the algorithm would be a far more contributory area of endeavour.

The presence of some very large hyperspheres also suggests the feasible region in this example is relatively concentrated, rather than evenly spread throughout the 7D space. This outcome is expected, as the three-chamber model has been shown to be practically identifiable when all outputs are known [9]. Equally, the number of feasible points is consistently <3.5% of the total number of points, which, as noted, is an interesting physiological model result, but also shows the concentration of feasible points is relatively low compared to infeasible points. This latter aspect is evident in the low slopes and rise of feasible points covered in Fig. 5.

When defining the feasible set, it is noteworthy the hyperspheres will tend to initially define the boundary of the set. Beginning on one edge of the feasible set, the first generated hypersphere will always have a radius of 1. Likewise, the second hypersphere will also

**Table 4**
Number of uniform points in a 7D discrete grid contained within a hypersphere of a given radius.

| Radius | Contained Points |
|---|---|
| 0 | 1 |
| 1 | 15 |
| 2 | 953 |
| 3 | 12,435 |
| 4 | 88,273 |
| 5 | 394,691 |
| 6 | 1,405,311 |
| 7 | 3,859,231 |
| 8 | 8,284,339 |
| 9 | 13,585,523 |
| 10 | 17,474,275 |

**Table 5**
Number of hyperspheres generated to define the infeasible space, and their radii for different numbers of input points, where 1 h is 3600s for reference.

| | Input Points | | | Generated Hyperspheres | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Total Points (P) in Table 2 | Total Feasible Points in Table 2 | MDUP Runtime | Number Generated | Maximum Radius | Minimum Radius | Mean Radius | Median Radius | Mode Radius |
| **Hyperspheres Defining Infeasible Space** | 162,000 | 6133 (3.79%) | 8.2 s | 2211 | 5.83 | 1 | 1.27 | 1 | 1 |
| | 279,936 | 9620 (3.44%) | 25.6 s | 3379 | 6.56 | 1 | 1.26 | 1 | 1 |
| | 823,543 | 28,811 (3.50%) | 179.9 s | 8592 | 7.84 | 1 | 1.23 | 1 | 1 |
| | 1,404,928 | 51,514 (3.67%) | 491.9 s | 13,917 | 8.19 | 1 | 1.21 | 1 | 1 |
| | 2,097,152 | 73,936 (3.53%) | 559.4 s | 19,053 | 9.22 | 1 | 1.21 | 1 | 1 |
| | 3,359,232 | 122,098 (3.63%) | 2403.8 s | 29,740 | 9.17 | 1 | 1.19 | 1 | 1 |
| | 4,782,969 | 168,259 (3.52%) | 4365.7 s | 38,478 | 9.85 | 1 | 1.18 | 1 | 1 |
| | 7,290,000 | 262,507 (3.60%) | 3.4 h | 56,866 | 12 | 1 | 1.17 | 1 | 1 |
| | 8,100,000 | 289,853 (3.58%) | 4.0 h | 60,983 | 10.91 | 1 | 1.17 | 1 | 1 |
| | 10,000,000 | 351,773 (3.52%) | 4.6 h | 71,271 | 11.36 | 1 | 1.17 | 1 | 1 |
| | 19,487,171 | 687,688 (3.53%) | 19.4 h | 126,608 | 12.92 | 1 | 1.16 | 1 | 1 |
| | 35,831,808 | 1,265,034 (3.53%) | 51.9 h | 210,433 | 13.75 | 1 | 1.15 | 1 | 1 |

**Table 6**
Number of hyperspheres generated to define the feasible space, and their radii for different numbers of input points, where 1 h is 3600s for reference.

| | Input Points | | | Generated Hyperspheres | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Total Points (P) in Table 2 | Total Feasible Points in Table 2 | MDUP Runtime | Number Generated | Maximum Radius | Minimum Radius | Mean Radius | Median Radius | Mode Radius |
| **Hyperspheres Defining Feasible Space** | 162,000 | 6133 (3.79%) | 8.9 s | 1926 | 1.41 | 1 | 1.01 | 1 | 1 |
| | 279,936 | 9620 (3.44%) | 22.7 s | 3023 | 1.73 | 1 | 1.01 | 1 | 1 |
| | 823,543 | 28,811 (3.50%) | 174.0 s | 8037 | 2 | 1 | 1.02 | 1 | 1 |
| | 1,404,928 | 51,514 (3.67%) | 491.4 s | 13,645 | 2.24 | 1 | 1.02 | 1 | 1 |
| | 2,097,152 | 73,936 (3.53%) | 1004.9 s | 18,981 | 2.24 | 1 | 1.02 | 1 | 1 |
| | 3,359,232 | 122,098 (3.63%) | 2586.7 s | 30,538 | 2.24 | 1 | 1.02 | 1 | 1 |
| | 4,782,969 | 168,259 (3.52%) | 4906.8 s | 40792 | 2.45 | 1 | 1.02 | 1 | 1 |
| | 7,290,000 | 262,507 (3.60%) | 3.0 h | 61,084 | 2.65 | 1 | 1.03 | 1 | 1 |
| | 8,100,000 | 289,853 (3.58%) | 3.7 h | 66,639 | 2.45 | 1 | 1.03 | 1 | 1 |
| | 10,000,000 | 351,773 (3.52%) | 5.4 h | 78,779 | 2.45 | 1 | 1.03 | 1 | 1 |
| | 19,487,171 | 687,688 (3.53%) | 18.3 h | 143,526 | 3 | 1 | 1.03 | 1 | 1 |
| | 35,831,808 | 1,265,034 (3.53%) | 59.1 | 246,027 | 3.16 | 1 | 1.03 | 1 | 1 |

always have a radius of 1, as it is centred on what must be another boundary point to be most distant. Consequently, the hyperspheres defining the feasible region, at least for this example, will often be predominantly very small-radius spheres, even more so than the infeasible region. Again, this outcome reflects the relatively far smaller number of feasible points.

More specifically, this tendency towards small-radius spheres can be seen in Fig. 5. When unsorted, the number of points added by each sphere is linear, for all numbers of input parameters. When sorted by radius, the generated curve is piecewise linear, indicating a small discrete number of possible point additions by each sphere. The noticeably smaller mean and maximum radii shown in Table 6 compared to the infeasible spheres shown in Table 5 reinforces that a higher proportion of spheres are smaller.

One benefit of defining the feasible set is the mean of the hypersphere centres can define a starting point for optimisation methods to improve their efficiency and reliability. Gradient descent methods are just one common example of optimisation algorithms which can improve in both runtime and reliability when given starting points more closely resembling the actual global minimums.

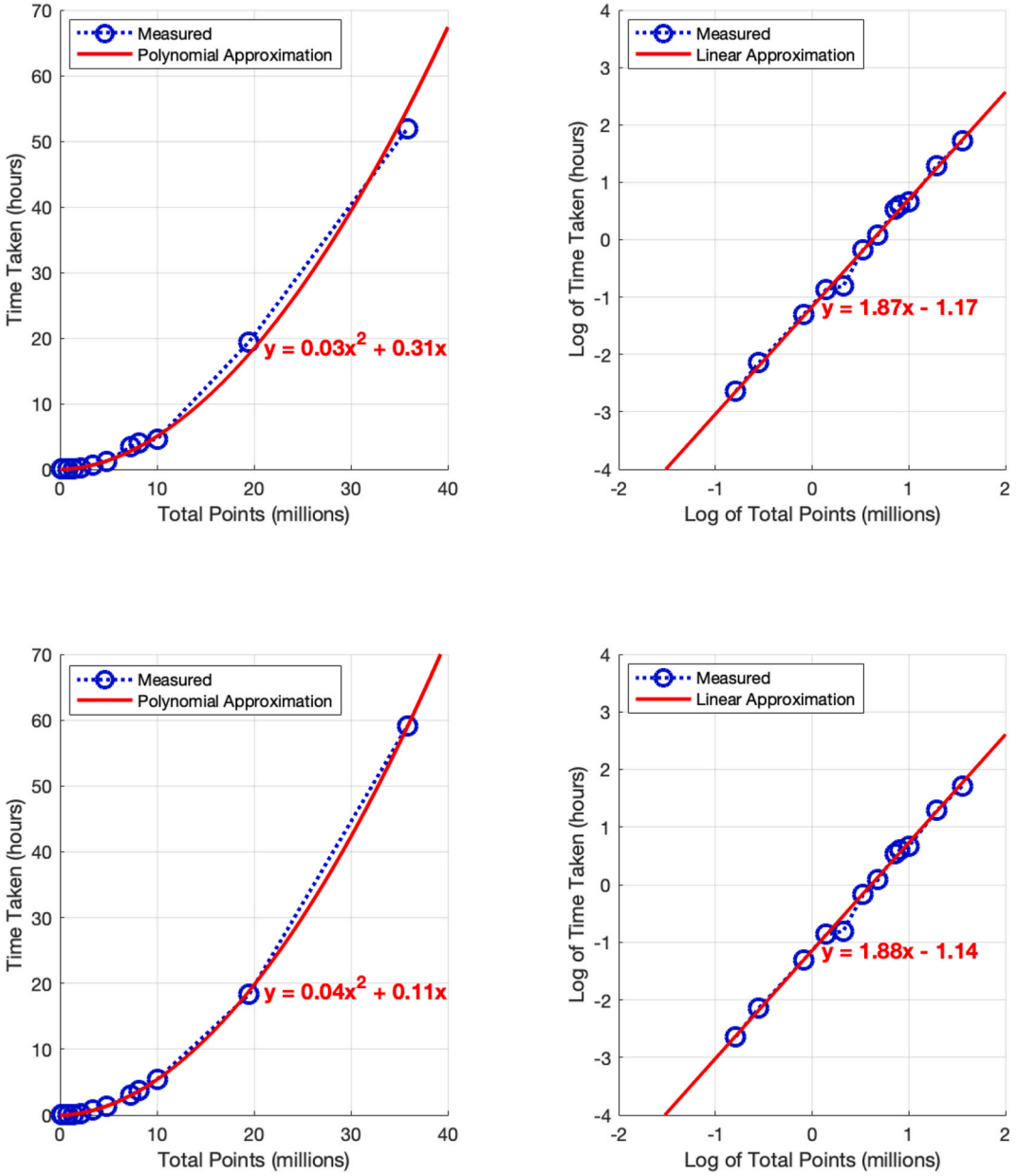



**Fig. 3.** Measured times to fill the infeasible space. Left: measured time taken for MDUP method to complete for different numbers of total points, with the corresponding polynomial to the linear approximation shown on loglog plot. Right: loglog of time taken for the MDUP method to complete with different numbers of input points, with a linear approximation. Top: defining the infeasible space. Bottom; defining the feasible space.

### *4.2. Computational effort and sensitivity analysis*

Fig. 3 (left) shows a strong quadratic behaviour over the range of data parameter sets or points examined for the infeasible region. The loglog graph in Fig. 3 (right) shows a strong linear relationship between the log of the total number of input points and the time taken for the algorithm to complete. This power law related behaviour matches expectations, and allows the computational time to be estimated, noting the use of compiled programming languages or parallelisation would add significant speed over the experimental computational set up used. Equally, the use of MATLAB on a standard desktop and Fig. 3 show relatively large numbers of 7D parameter sets can be examined in one day or less. Adding more RAM might also speed up the computational time as processing power was not 100% utilised during these case runs.

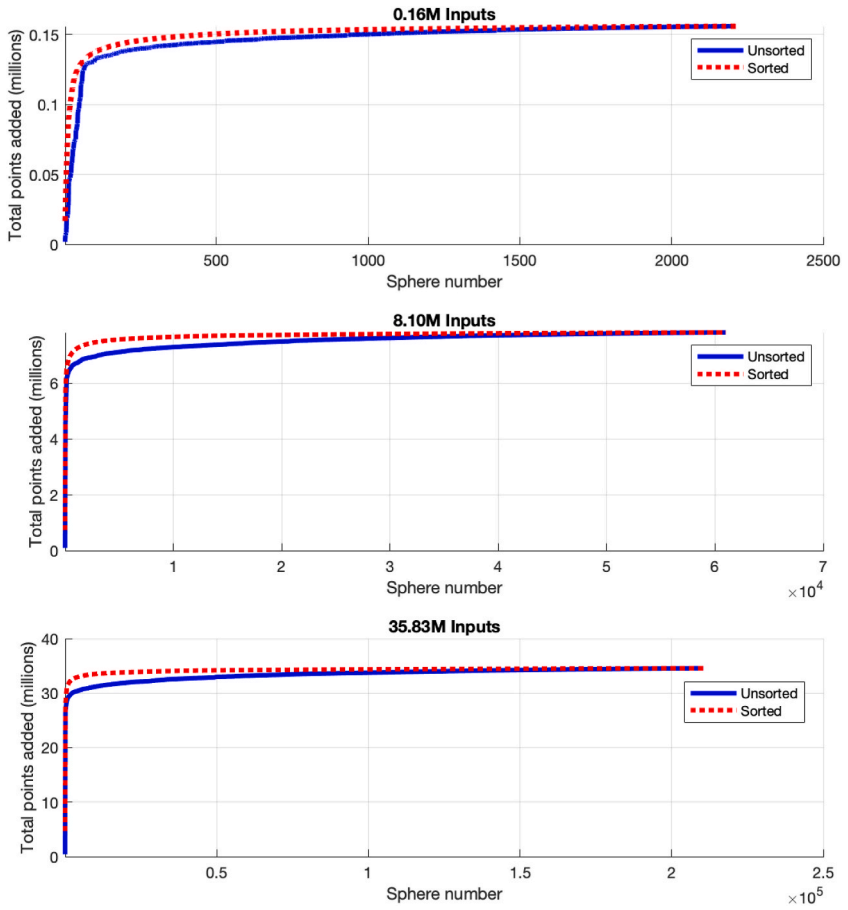



**Fig. 4.** Infeasible space: Cumulative sum of the number of covered points added by each hypersphere, when defining the infeasible space, in order of generation (solid blue), and cumulative sum of the number of covered points added by each hypersphere, in descending order of radius (dotted red). Each plot represents a test of the algorithm with different numbers of inputs.

### *4.3. Method variants*

Some alternative hypersphere methods were considered, but each had weaknesses not held by the MDUP method. These variants are delineated to show the potential limitations or advantages of the method as presented in the Methods. Following discussion will focus on filling the infeasible space, but the method is identical when filling the feasible space.

A **Complete Implementation** variant iterates through all infeasible points and calculates the minimum Euclidean distance from each of those points to the feasible set. This approach fills the entire infeasible space with hyperspheres. The hyperspheres are then sorted in descending order of radius, and then checked in order to see whether they cover additional points. These steps are outlined.

1. For every infeasible point, calculate the Euclidean distance to the nearest point in the feasible set, creating a hypersphere for every infeasible point. At this stage the points covered by the hypersphere are not considered.
2. Mark each point in the infeasible space as ‘uncovered’.
3. Sort the hyperspheres in descending order of radius.
4. For each sorted hypersphere, check if it contains any ‘uncovered’ points. If so, mark those points as ‘covered’ and remove them from the list of remaining uncovered infeasible points. If the hypersphere does not contain any uncovered points, remove it from the list of hyperspheres.

Just as with the presented MDUP method, the result of this process is a collection of hyperspheres which cover the entire infeasible space. However, this approach begins by generating a hypersphere for all points in the infeasible space. The comprehensiveness this provides may provide additional insight in some applications, and allows a user to search for all possible largest hyperspheres with certainty of finding them.

This method is extremely consistent and allows for each hypersphere to easily be considered individually. In addition, because the creation of each hypersphere is independent in this method, the complete method is the only method able to be easily implemented

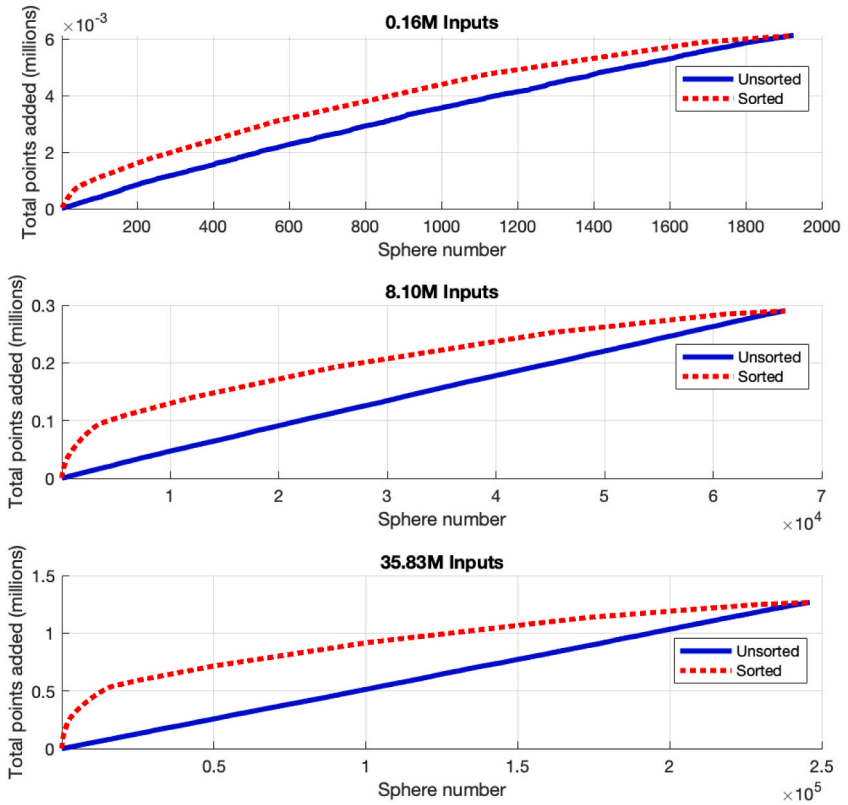



**Fig. 5.** Feasible space. Cumulative sum of the number of covered points added by each hypersphere, when defining the feasible space, in order of generation (solid blue), and cumulative sum of the number of covered points added by each hypersphere, in descending order of radius (dotted red). Each plot represents a test of the algorithm with different numbers of inputs.

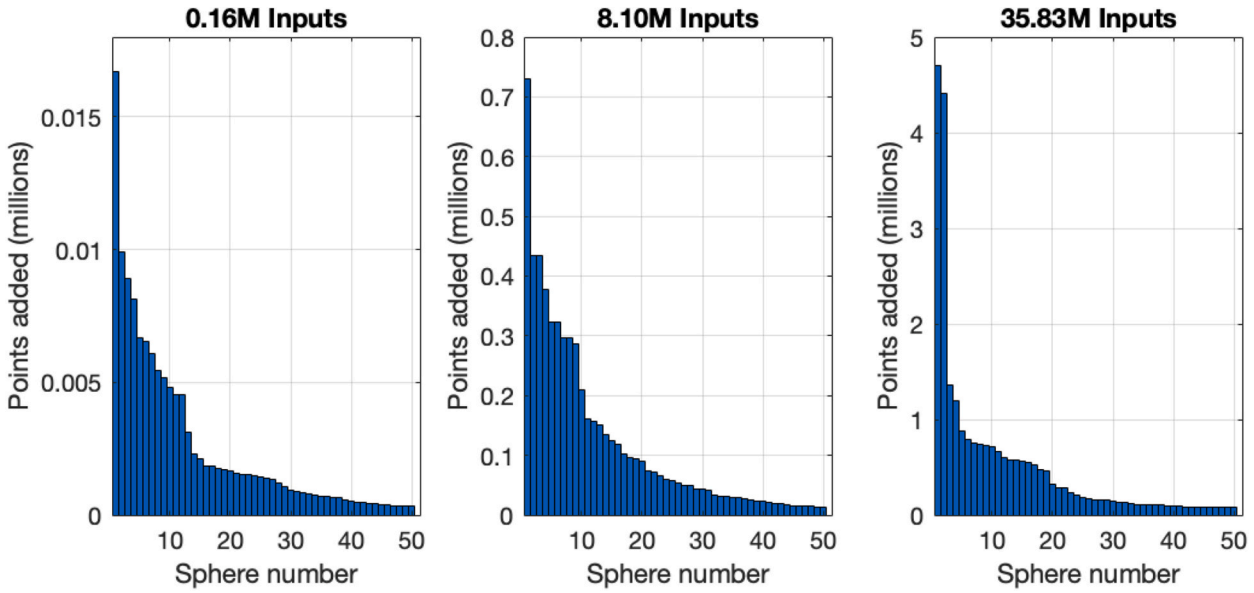



**Fig. 6.** Infeasible space. Number of covered points, when defining the infeasible space, added by the 50 largest-radius hyperspheres, in descending order of radius. From left to right: the algorithm with 0.16, 8.10, and 35.83 million total input points.

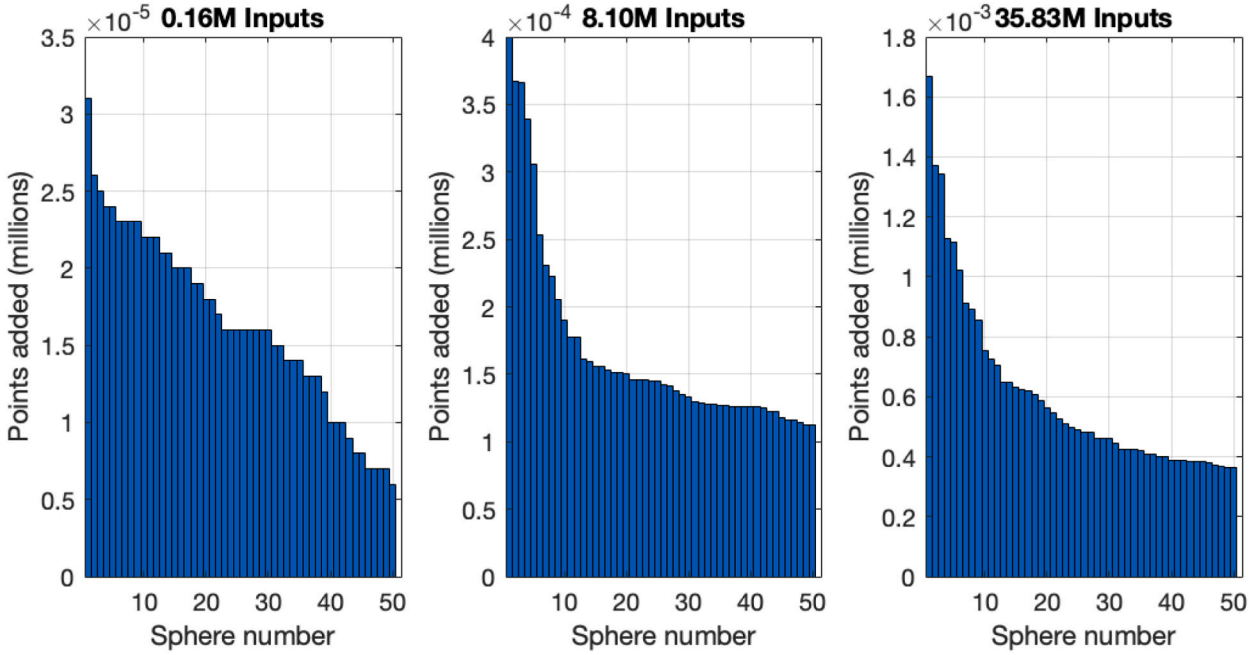



**Fig. 7.** Feasible space. Number of covered points, when defining the feasible space, added by the 50 largest-radius hyperspheres, in descending order of radius. From left to right: the algorithm with 0.16, 8.10, and 35.83 million total input points.

**Table 7**
Centre points of largest 10 spheres for 0.16 million input points (Infeasible). Parameter values are shown as their true (prior to scaling) value.

| Input Parameter | Sphere Number | | | | | | | | | | Sphere Centre Means | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Largest 10 Spheres | All Spheres |
| $E_h$ (mmHg ml$^{-1}$) | 3.00 | 1.80 | 0.60 | 1.20 | 3.00 | 3.00 | 0.60 | 0.60 | 3.00 | 3.00 | 1.98 | 1.68 |
| $E_a$ (mmHg ml$^{-1}$) | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 1.20 | 3.00 | 3.00 | 1.14 | 1.62 |
| $E_v$ (mmHg ml$^{-1}$) | 0.50 | 0.10 | 0.10 | 0.10 | 0.50 | 0.10 | 0.50 | 0.10 | 0.10 | 0.10 | 0.22 | 0.33 |
| $R_i$ (mmHg s ml$^{-1}$) | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.00 | 0.50 |
| $R_o$ (mmHg s ml$^{-1}$) | 0.05 | 0.10 | 0.10 | 0.02 | 0.10 | 0.02 | 0.03 | 0.10 | 0.02 | 0.10 | 0.06 | 0.05 |
| $R_c$ (mmHg s ml$^{-1}$) | 0.3 | 2.0 | 0.3 | 0.3 | 0.3 | 2.0 | 2.0 | 2.0 | 0.3 | 0.3 | 1.00 | 1.17 |
| SBV (ml) | 200 | 33 | 133 | 33 | 33 | 33 | 33 | 200 | 33 | 33 | 76.67 | 116.67 |
| Radius | 5.83 | 5.74 | 5.66 | 5.48 | 5.48 | 5.10 | 5.10 | 4.80 | 4.69 | 4.69 | | |
| No. points added | 16,686 | 9900 | 8905 | 8115 | 6674 | 6556 | 6065 | 5423 | 5166 | 4822 | | |

with parallel computing processes. Unfortunately, due to the large number of generated hyperspheres this method is extremely computationally time-consuming (taking over 560 s to complete for 162,000 input points – more than 68 times slower than the MDUP method) even when utilising parallelisation, making it incredibly expensive for large high-dimensional data sets.

To reduce computational load lower than the MDUP method, the **Next Uncovered Point (NUP)** variant functions similarly to the MDUP method, but circumvents the distance calculations required to find the most distant uncovered point. Instead, this method iterates through all points in the same manner as the Complete Implementation variant, but skips points which are covered by an existing hypersphere. It is defined.

1. Start with a corner point in the infeasible set.
2. Calculate the Euclidean distance to the nearest point in the feasible set to find the radius of the hypersphere centred on that point.
3. Mark all points contained within this hypersphere as ‘covered’ and remove them from the remaining list of uncovered infeasible points.
4. In sequence move through infeasible points from the current centre point until one is reached that is uncovered. This will be the centre point of the next hypersphere.
5. Repeat steps 2–4 until all infeasible points are marked as covered.

This variant struggles to create spheres once it moves ‘above’ the feasible set. Its inability to select distant points creates a layering effect of hyperspheres, which significantly hinders its performance. Continually updating which points are covered and uncovered also means this variant is not easily parallelised.

**Table 8**
Centre points of largest 10 spheres for 8.10 million input points (Infeasible). Parameter values are shown as their true (prior to scaling) value.

| Input Parameter | Sphere Number | | | | | | | | | | Sphere Centre Means | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Largest 10 Spheres | All Spheres |
| $E_h$ (mmHg ml$^{-1}$) | 0.33 | 0.33 | 3.00 | 0.33 | 3.00 | 0.33 | 3.00 | 3.00 | 0.33 | 0.33 | 1.40 | 1.67 |
| $E_a$ (mmHg ml$^{-1}$) | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 3.00 | 0.33 | 0.33 | 0.33 | 2.00 | 0.77 | 1.63 |
| $E_v$ (mmHg ml$^{-1}$) | 0.40 | 0.10 | 0.05 | 0.50 | 0.05 | 0.05 | 0.50 | 0.50 | 0.05 | 0.50 | 0.27 | 0.24 |
| $R_i$ (mmHg s ml$^{-1}$) | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.4 | 1.0 | 0.94 | 0.55 |
| $R_o$ (mmHg s ml$^{-1}$) | 0.10 | 0.01 | 0.10 | 0.10 | 0.01 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.08 | 0.06 |
| $R_c$ (mmHg s ml$^{-1}$) | 2.0 | 2.0 | 2.0 | 0.2 | 0.2 | 0.2 | 0.2 | 2.0 | 0.2 | 0.2 | 0.92 | 1.10 |
| SBV (ml) | 20 | 20 | 20 | 200 | 20 | 20 | 20 | 200 | 20 | 20 | 56.00 | 110.00 |
| Radius | 10.91 | 10.77 | 10.34 | 10.20 | 10.10 | 9.95 | 9.38 | 9.33 | 9.11 | 9.06 | | |
| No. points added | 730,271 | 435,230 | 434,422 | 377,523 | 323,675 | 323,275 | 297,336 | 296,309 | 286,810 | 210,173 | | |

**Table 9**
Centre points of largest 10 spheres for 35.85 million input points (Infeasible). Parameter values are shown as their true (prior to scaling) value.

| Input Parameter | Sphere Number | | | | | | | | | | Sphere Centre Means | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Largest 10 Spheres | All Spheres |
| $E_h$ (mmHg ml$^{-1}$) | 0.25 | 0.25 | 3.00 | 0.25 | 0.25 | 3.00 | 0.25 | 2.25 | 3.00 | 3.00 | 1.55 | 1.43 |
| $E_a$ (mmHg ml$^{-1}$) | 0.25 | 0.75 | 0.25 | 0.25 | 0.25 | 0.25 | 3.00 | 0.25 | 0.25 | 0.25 | 0.57 | 4.40 |
| $E_v$ (mmHg ml$^{-1}$) | 0.04 | 0.50 | 0.04 | 0.04 | 0.50 | 0.50 | 0.04 | 0.50 | 0.50 | 0.25 | 0.29 | 0.27 |
| $R_i$ (mmHg s ml$^{-1}$) | 0.9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.99 | 0.54 |
| $R_o$ (mmHg s ml$^{-1}$) | 0.01 | 0.10 | 0.10 | 0.10 | 0.10 | 0.01 | 0.08 | 0.10 | 0.10 | 0.01 | 0.07 | 0.05 |
| $R_c$ (mmHg s ml$^{-1}$) | 2.0 | 0.2 | 1.3 | 0.2 | 1.3 | 0.2 | 0.2 | 2.0 | 2.0 | 0.2 | 0.95 | 0.98 |
| SBV (ml) | 17 | 17 | 17 | 200 | 200 | 200 | 17 | 17 | 200 | 17 | 90.00 | 106.67 |
| Radius | 13.75 | 13.11 | 13 | 12.57 | 12.17 | 12.12 | 12.04 | 11.83 | 11.27 | 11.27 | | |
| No. points added | 4,705,625 | 4,412,300 | 1,361,791 | 1,190,464 | 878,193 | 791,349 | 750,731 | 739,557 | 729,528 | 721,374 | | |

### 4.4. Limitations

Each iteration of the MDUP method relies on the updated list of covered and uncovered points from the previous iteration. The method is thus not easily parallelised. This issue significantly slows its runtime, and negates much of the advantage of modern computing devices. As noted, the use of C or a computer with more RAM could add significant computation speed, where compiled C code is often 20-100x faster than MATLAB.

The MDUP method can generate different results based on the selected starting point. Although the distribution of radii and number of spheres tends to be minimally affected as a whole, different starting points will generate completely different sets of hyperspheres to cover the infeasible space. Thus, there may, for some problem sets be specific advantage in any first assessment of a larger feasible or infeasible region starting point. This outcome highlights the point that while the method is general, the outcomes are problem specific.

Both the MDUP method and its Complete Implementation variant are currently not optimised. Different programming languages, increased computing power, and superior programmers could certainly improve performance.

Finally, in this work the method was successively run to observe the influence of additional input points on computational load and runtime. In practice, this method would serve as a one-time computation with an application-dependent granularity, and researchers may decide to prioritise the comprehensiveness of the complete implementation in exchange for a longer runtime.

### 4.5. Comparative analysis to accepted methods

The MDUP hypersphere method is able to provide information about the structure of data in its full dimensional state, making it superior to PCA and t-SNE analyses in this regard. It also does not rely on a user-defined number of centroids, making it less user-dependent than k-means clustering. The MDUP method also scales readily into any number of dimensions, does not require the parameter space to align in specific ways, and is geometrically simple, creating advantages over glyph-based visualisations, hyper-slicers, and region-growing algorithms respectively.

## 5. Conclusions

A novel method for analysing the structure and shape of high-dimensional points using hyperspheres was introduced, in the context of analysing physiologically feasible inputs to the three-chamber cardiovascular model. The method is able to provide information about the distribution and structure of points in a high-dimensional space, without requiring user input or sacrificing information through dimensionality reduction. The full significance of the specific groupings and structure of the hypersphere is not yet entirely understood, but could contain further insights into the structure of high-dimensional point sets. An understanding of the implications of sphere location and size could assist in initial optimisation method parameters. Finally, while the algorithm is general to any size space, the results here are specific to this problem case, where this research also presents both method variants, as well as suggesting the need to parallelise the algorithm.

## CRediT authorship contribution statement

**Nicolas A.C. Davey:** Software, Writing – original draft, Investigation. **J. Geoffrey Chase:** Conceptualization, Funding acquisition, Supervision, Writing – review & editing. **Cong Zhou:** Supervision, Writing – review & editing. **Liam Murphy:** Writing – review & editing.

## 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.

## Acknowledgements

This work was supported by the NZ National Science Challenge 7, Science for Technology and Innovation (2019-S3-CRS). The authors also acknowledge support from the EU H2020 R&I programme (MSCA-RISE-2019 call) under grant agreement #872488 — DCPM; and the EU H2020 ERA Permed JTC2021, “Personalised perfusion guided fluid therapy”.

## References

[1] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[2] G. Hinton, S. Roweis, Stochastic Neighbor Embedding, vol. 15, 2003, 06/20.
[3] A. Inselberg, The plane with parallel coordinates, Vis. Comput. 1 (2) (2023) 69–91, https://doi.org/10.1007/bf01898350.
[4] J. Wang, X. Liu, H.W. Shen, G. Lin, Multi-resolution climate ensemble parameter analysis with nested parallel coordinates plots, IEEE Trans. Visual. Comput. Graph. 23 (1) (2017) 81–90, https://doi.org/10.1109/TVCG.2016.2598830.
[5] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, T. Moller, Visual parameter space analysis: a conceptual framework, IEEE Trans. Visual. Comput. Graph. 20 (2014) 2161–2170, https://doi.org/10.1109/TVCG.2014.2346321, 12/31.
[6] A. Bock, A. Pembroke, M.L. Mays, L. Rastaetter, T. Ropinski, A. Ynnerman, Visual verification of space weather ensemble simulations, in: 2015 IEEE Scientific Visualization Conference (SciVis), 2015, pp. 17–24, https://doi.org/10.1109/SciVis.2015.7429487, 25-30 Oct. 2015.

[7] M. Evers, L. Linsen, Multi-dimensional parameter-space partitioning of spatio-temporal simulation ensembles, Comput. Graph. 104 (2022) 140–151, https://doi.org/10.1016/j.cag.2022.04.005, 2022/05/01/.
[8] C. Revol-Muller, T. Grenier, T. Li, H. Benoit-Cattin, Feature space region growing, in: *2012 19th IEEE International Conference On Image Processing*, 30 Sept.-3 Oct. 2012, 2012, pp. 2585–2588, https://doi.org/10.1109/ICIP.2012.6467427.
[9] A. Pironet, P.D. Docherty, P.C. Dauby, J.G. Chase, T. Desaive, Practical identifiability analysis of a minimal cardiovascular system model, Comput. Methods Progr. Biomed. 171 (2019) 53–65, https://doi.org/10.1016/j.cmpb.2017.01.005, 2019/04/01/.
[10] L. Murphy, S. Davidson, J.G. Chase, J.L. Knopp, T. Zhou, T. Desaive, Patient-specific monitoring and trend analysis of model-based markers of fluid responsiveness in sepsis: a proof-of-concept animal study, Ann. Biomed. Eng. 48 (2) (2020) 682–694, https://doi.org/10.1007/s10439-019-02389-9, 2020/02/01.
[11] J. Cushway, L. Murphy, J.G. Chase, G.M. Shaw, T. Desaive, Physiological trend analysis of a novel cardio-pulmonary model during a preload reduction manoeuvre, Comput. Methods Progr. Biomed. 220 (2022) 106819, https://doi.org/10.1016/j.cmpb.2022.106819, 2022/06/01/.
[12] A. Pironet, T. Desaive, J. Geoffrey Chase, P. Morimont, P.C. Dauby, Model-based computation of total stressed blood volume from a preload reduction manoeuvre, Math. Biosci. 265 (2015) 28–39, https://doi.org/10.1016/j.mbs.2015.03.015, 2015/07/01/.
[13] P.J. Hunter, N.P. Smith, The cardiac physiome project, J. Physiol. 594 (23) (Dec 1 2016) 6815–6816, https://doi.org/10.1113/jp273415 (in eng).
[14] S. Safaei, et al., Roadmap for cardiovascular circulation model, J. Physiol. 594 (23) (Dec 1 2016) 6909–6928, https://doi.org/10.1113/jp272660 (in eng).
[15] C. Starfinger, et al., Model-based identification and diagnosis of a porcine model of induced endotoxic shock with hemofiltration, Math. Biosci. 216 (2) (Dec 2008) 132–139, https://doi.org/10.1016/j.mbs.2008.08.014 (in eng).
[16] T. Desaive, O. Horikawa, J. Ortiz, J. Chase, Model-based management of cardiovascular failure: where medicine and control systems converge, Annu. Rev. Control 48 (2019), https://doi.org/10.1016/j.arcontrol.2019.05.003, 05/01.
[17] J.G. Chase, et al., Next-generation, personalised, model-based critical care medicine: a state-of-the art review of in silico virtual patient models, methods, and cohorts, and how to validation them, Biomed. Eng. Online 17 (1) (2018) 24, https://doi.org/10.1186/s12938-018-0455-y, 2018/02/20.
[18] A. Pironet, P.C. Dauby, J.G. Chase, P.D. Docherty, J.A. Revie, T. Desaive, Structural identifiability analysis of a cardiovascular system model, Med. Eng. Phys. 38 (5) (2016) 433–441, https://doi.org/10.1016/j.medengphy.2016.02.005, 2016/05/01/.
[19] T. Desaive, O. Horikawa, J.P. Ortiz, J.G. Chase, Model-based management of cardiovascular failure: where medicine and control systems converge, Annu. Rev. Control 48 (2019) 383–391, https://doi.org/10.1016/j.arcontrol.2019.05.003, 2019/01/01/.
[20] C. Zhou, J.G. Chase, G.W. Rodgers, Support vector machines for automated modelling of nonlinear structures using health monitoring results, Mech. Syst. Signal Process. 149 (2021) 107201, https://doi.org/10.1016/j.ymssp.2020.107201, 2021/02/15/.
[21] Y. Chen, et al., Machine learning model identification and prediction of patients' need for ICU admission: a systematic review, in English, Am. J. Emerg. Med. 73 (Nov 2023) 166–170, https://doi.org/10.1016/j.ajem.2023.08.043, 2023-10-30 2023.
[22] P.C. Lin, K.T. Chen, H.C. Chen, M.M. Islam, M.C. Lin, Machine learning model to identify sepsis patients in the emergency department: algorithm development and validation, J. Personalized Med. 11 (11) (Oct 21 2021), https://doi.org/10.3390/jpm11111055 (in eng).
[23] L.F. Murphy, Clinically Feasible Model-Based Methods to Guide Cardiovascular Fluid Therapy in the ICU : a Thesis Submitted for the Degree of Doctor of Philosophy in Mechanical Engineering at the University of Canterbury, Christchurch, New Zealand, [University of Canterbury], Christchurch, New Zealand, 2022 [Online]. Available: https://hdl.handle.net/10092/104774.
[24] A. Pironet, et al., "Model-Based decision support algorithm to guide fluid Resuscitation**This work was supported by the French community of Belgium, the Belgian funds for scientific research (F.R.S.-FNRS) and EU marie curie actions (FP7-PEOPLE-2012-IRSES),", IFAC-PapersOnLine 49 (5) (2016) 224–229, https://doi.org/10.1016/j.ifacol.2016.07.117, 2016/01/01/.
[25] J. Cushway, L. Murphy, J.G. Chase, G.M. Shaw, T. Desaive, "Modelling patient specific cardiopulmonary interactions,", Comput. Biol. Med. 151 (Pt A) (Dec 2022) 106235, https://doi.org/10.1016/j.compbiomed.2022.106235 in eng.