University of Strasbourg

Master 1 CSMI

Implementation and comparison of cardiovascular models

Colin Holler

Supervisor : M. Prud'homme Christophe

Academic year 2020-2021

Contents

1 Introduction.

1.1 General context.

This internship is part of the first year of the CSMI master's degree. From June 1 to July 31, I did my internship at the maths laboratory of the University of Strasbourg in teleworking. This internship was an opportunity for me to broaden my knowledge on subjects such as the cardiovascular field. This internship also allowed me to understand different existing models. The topic I worked on was a direct continuation of the group project done in the first year. In this report we will find several parts of the project report.

1.2 Company presentation.

This internship is a direct continuation of the project with **Groupama FDJ**, so a presentation of this company is necessary. At the beginning of the 1997 season, thanks to the passion of Marc and Yvon Madiot, the cycling team "Française des Jeux" was born. 2018 marks a turning point in the team's history, with the arrival of Groupama, the new title partner. The association of these two major players in French sport gives birth to the **Groupama-FDJ Cycling Team** [8].

Figure 1: Groupama-FDJ logo (Source : Groupama-FDJ Equipe Cycliste [8])

The current team leaders are Thibault Pinot (climber) and Arnaud Démarre (sprinter). In 24 years, the team has won more than 500 victories, including 11 French Champion jerseys, 3 cycling Monuments and 44 stages of the Grand Tours. The team is part of the UCI WorldTour and therefore among the 19 best teams in the world. In particular, in 2020 it was the 9th world team and the 1st French team. Moreover, recently David Gaudu, a climber from Groupama-FDJ, won the sixth and last stage of the Tour of the Basque Country.

1.3 Subjects explanation.

The idea of the internship is to understand in particular a specific system proposed by the **Pulse Physiology Engine** platform. Indeed, in our project, we are interested in several models such as the nutrition, environment and cardiovascular model. In our case, we will deepen our research on the cardiovascular model. We will go into the implementation of this model and understand what is called the Data Flow. Then we are interested in another model, that of the brain-eye coupling, which can be similar to the one of **Pulse**.

1.4 Project management.

I mainly worked on my own by searching the website associated with **Pulse Physiology Engine** as well as reading the articles given by my supervisor Mr. Christophe Prud'homme who is a teacher-researcher at the University of Strasbourg. I also contacted my project mates Melissa and Jimmy for some help on how to use Pulse.

2 Pulse

2.1 Pulse Physiology Engine framework.

The **Pulse Physiology Engine** is a C++ based, comprehensive human physiology simulator that drives medical education, research, and training technologies. The engine enables accurate and consistent **physiology simulation** across the medical community.

Figure 2: Pulse Physiology Engine logo (Source : Pulse [1])

The engine is a **body physiology model** that combines **physics-based parameter models** and **control system mechanisms** to model real-time system level physiologic behaviors. Parameter models use electrical circuit analogs to represent the major physiologic systems. Closed loop circuits can be defined easily within the Common Data Model [2]. It is a specification of all data and relationships associated with the software [2]. It provides a data exchange format between models of physiological engine, but also between any application and physiological engine. Since the language is common to all, this makes it easier to implement new models in the application. Closed loop circuits fully dynamic, and can be manipulated and modified at each time step. The solver can be used to analyze electrical, fluid, and thermal circuits using a unit in the international system. The solver outputs were validated using the outputs from an existing known third party circuit solver. All results matched the validation data, confirming this is a sound approach for the engine. Here with the environment system below, the Ambient node contains both thermal properties (temperature), fluid properties (pressure), and substance properties (volume fraction, aerosol concentration). It is assigned as the Respiratory and Anesthesia Machine circuit's reference node, and therefore, interacts with them directly. Any changes to the Ambient node properties automatically propagates through the other systems. We will see later an example of an electrical circuit.

Pulse has been developed through a combination of funding by many government and private entities, and is a significantly improved and extended fork from the DoD-funded BioGears program. The Pulse repository is maintained by the Kitware team that includes the original core BioGears creators. Pulse has recently been incorporated into a number of commercials, research, and academic tools for medical simulation. Moreover, models and results are validated using a combination of peer-reviewed publications and subject expertise.

2.2 First steps with Pulse.

The engine is driven by a set of instructions written in a scenario file. The structure of a scenario must respect the Common Data Model.
The scenario is stored in a JSON file and contains the following execution information ’":

- Patient File and optional conditions.
- A list of values to return from the engine.
- A list of actions to execute, for example a cardiac arrest during the simulation.

As described in Listing **1**, the JSON file starts with a name and description, but this is not used in execution.

```
{
"Name": "Vitals Monitor",
"Description": "Data associated with a vitals monitor.",
...
```

Listing 1: Begin of VitalsMonitor.json.

Then, it is necessary to specify which file is used to initialize the patient’s data. At the time of the simulation, this data is checked to ensure that the patient is stable before any action is taken. Below, the *ExerciseEnvironment.json* file initializes the data from the patient’s environment.

```
...
"PatientConfiguration": {
  "PatientFile": "StandardMale.json",
  "Conditions": { "AnyCondition": [{
    "EnvironmentCondition": {
      "InitialEnvironmentalConditions": {
        "EnvironmentalConditionsFile": "./environments/ExerciseEnvironment.json"
        }
      }
    },{
      "PatientCondition" : {
        "ConsumeMeal": { "Meal": { "Nutrition": {
          "Carbohydrate": {
            "ScalarMass": { "Value": 200.0, "Unit": "g" }
          },
          "CarbohydrateDigestionRate": {
            "ScalarMassPerTime": { "Value": 1.083, "Unit": "g/min" }
          },
          "Fat": {
            "ScalarMass": { "Value": 20.0, "Unit": "g" }
          },
          "FatDigestionRate": {
            "ScalarMassPerTime": { "Value": 0.055, "Unit": "g/min" }
          },
          "Protein": {
            "ScalarMass": { "Value": 30.0, "Unit": "g" }
          },
          "ProteinDigestionRate": {
            "ScalarMassPerTime": { "Value": 0.071, "Unit": "g/min" }
          },
          "Calcium": {
            "ScalarMass": { "Value": 521, "Unit": "mg" }
          },
          "Sodium": {
            "ScalarMass": { "Value": 1604, "Unit": "mg" }
          },
          "Water": {
            "ScalarVolume": { "Value": 1.0, "Unit": "L"}
          }},
        "ElapsedTime": { "ScalarTime": { "Value": 2.0, "Unit": "hr"}}}}}}
      ]}},
...
```

Listing 2: Configuration of initial data.

Listing **3** describes Data Requests that correspond to the output data after simulation, stored in a CSV file. Currently, there are four supported types of data requests : Physiology System Data, Environment Data, Substance Data, Equipment Data and Compartment Data. For each data, we have to choose the precision of the rounded values and give the unit of measurement.

```
...
"DataRequestManager": {
  "DataRequest": [
    {
      "DecimalFormat":{"Precision":2}, "Category":"Physiology",
      "PropertyName":"HeartRate", "Unit":"1/min"
    },
    {
      "DecimalFormat":{"Precision":3}, "Category":"Physiology",
      "PropertyName":"OxygenSaturation", "Unit":"unitless"
    },
    {
      "DecimalFormat":{"Precision":1}, "Category":"Physiology",
      "PropertyName":"SkinTemperature", "Unit":"degC"
    },
  ]},
...
```

Listing 3: Data requests for outputs.

Then, conditions give instructions to the engine to apply certain changes to the engine to simulate the specified conditions. We can change the environment completely by including a file that stores this data, or we can change all environment settings manually, as in Listing **4**. To change the amount of nutrients consumed, you can also change all parameters, as in the example.

```
...
"AnyAction": [{
  "AdvanceTime": {
    "Time": {
      "ScalarTime": {
        "Value": 50.0,
          "Unit": "s"
        }
      }
    }
  },
  { "EnvironmentAction": {
      "ChangeEnvironmentalConditions": {
        "EnvironmentalConditionsFile": "./environments/Hypobaric4000m.json"
      }
    }
  },
  { "PatientAction": {
      "ConsumeNutrients": {
        "Nutrition": {
          "Carbohydrate": {
            "ScalarMass": { "Value":415.0, "Unit":"g" }
          },
          "CarbohydrateDigestionRate": {
            "ScalarMassPerTime": { "Value":0.5, "Unit":"g/min" }
          },
          "Fat": {
            "ScalarMass": { "Value":83.0, "Unit":"g" }
          },
          "FatDigestionRate": {
            "ScalarMassPerTime": { "Value":0.055, "Unit":"g/min" }
          },
          "Protein": {
            "ScalarMass": { "Value":99.6, "Unit":"g"}
          },
          "ProteinDigestionRate": {
            "ScalarMassPerTime": { "Value":0.071, "Unit":"g/min" }
          },
          "Calcium": {
            "ScalarMass": { "Value":1.0, "Unit":"g"}
          },
          "Sodium": {
            "ScalarMass": { "Value":1.5, "Unit":"g"}
          },
          "Water": {
            "ScalarVolume": { "Value":3.7, "Unit":"L"}
          }
        }
      }
    }
  }
  ]
```

Listing 4: Patient actions and environment actions.

In our project we used the exercise action which consists of increase the patient's metabolic rate leading to an increase in core temperature, cardiac output, respiration rate and tidal volume. This will therefore allow us to model an effort of a cyclist. But in our case, we have several actions available to the cardiovascular system :

- **Cardiac Arrest** : The cardiac arrest event is a cessation of all cardiac and respiratory function. The cardiac arrest event can be triggered by systems of the engine when physiological thresholds are reached.
- **Hemorrhage** : Hemorrhage is a significant reduction in blood volume, which triggers a physiological response to stabilize cardiovascular function. Hemorrhage causes a reduction in the filling pressure of the circulation, resulting in a decrease in venous return. This is evidenced by a decrease in mean arterial pressure and cardiac output.
- **Pericardial Effusion** : The pericardial effusion action is used to model acute pericardial effusion by adding a flow source on the pericardium. This action leads to a volume accumulation over the course of the simulation. The accumulated volume is used to calculate a pressure source that is applied to the left and right heart.
- **Cardiopulmonary Resuscitation** : Chest compressions are simulated in the engine by adjusting the value of the pressure source that is connected between the common equipotential node and the capacitors that represent the right and left cardiac compliance. The pressure source represents the intrathoracic pressure, thus a positive pressure value represents an increase in intrathoracic pressure. The pressure on the source is calculated from the input force and the biometrics of the patient.

- **Exacerbations** : The pulmonary shunt condition has an exacerbation actions defined to allow for increased/decreased severities during runtime. This exacerbation action will instantaneously (i.e., during the simulation/scenario runtime) change the value of the pulmonary shunt (right, left, or both), based on the severity provided. Exacerbations can either degrade or improve the patient's current condition.
- **Hemorrhage Cessation** : A "tourniquet" may be applied to a bleeding distal portion of the body to reduce blood flow entering that portion and effectively stopping the bleeding. With the bleeding managed, vital signs should return to their normal state after a sufficient period of time. A "tourniquet" may be simulated in the engine by reducing the bleeding rate associated with a specified hemorrhage

2.3 Parameters description.

We have several parameters available to set up a patient. In our case, we will just present the descriptions of the **patient file**, the others like **environment file** and **nutrition file** are not interesting for us. In particular, the patient file contains all the parameters that describe his state of health.

```json
{
  "Name": "StandardMale",
  "Age": {
    "ScalarTime": {
      "Value": 44.0,
      "Unit": "yr"
    }
  },
  "Weight": {
    "ScalarMass": {
      "Value": 170.0,
      "Unit": "lb"
    }
  },
  "Height": {
    "ScalarLength": {
      "Value": 71.0,
      "Unit": "in"
    }
  },
  "BodyFatFraction": {
    "Scalar0To1": {
      "Value": 0.21
    }
  },
  "DiastolicArterialPressureBaseline": {
    "ScalarPressure": {
      "Value": 73.5,
      "Unit": "mmHg"
    }
  },
  "SystolicArterialPressureBaseline": {
    "ScalarPressure": {
      "Value": 114.0,
      "Unit": "mmHg"
    }
  }
}
```

Listing 5: Initial Patient data

In Listing **5**, we describe how to implement this data.

Age : Age of the cyclist in **years**.

Weight : Weight of the cyclist in pound-mass, **lb**.

Height : Height of the cyclist in inches, **in**.

BodyFatFraction : Number between 0 and 1 associated with body fat.

DiastolicArterialPressureBaseline : Blood pressure is traditionally measured using auscultation with a mercury-tube sphygmomanometer. The diastolic blood pressure is the minimum pressure experienced in the aorta when the heart is relaxing before ejecting blood into the aorta from the left ventricle, the unit is **mmHg** and generally the value is approximated at 80 mmHg.

HeartRateBaseline : Heart rate is the speed of the heartbeat measured by the number of contractions (beats) of the heart per minute. The unit is in Hertz i.e. **1/min**.

RespirationRateBaseline : The respiratory rate is the rate at which breathing occurs. The unit is also in Hertz i.e. **1/min**.

SystolicArterialPressureBaseline : Systolic pressure refers to the maximum pressure within the large arteries when the heart muscle contracts to propel blood through the body, the unit is therefore **mmHg**.

2.4 Using Pulse.

2.4.1 How to configure an exercise scenario.

Before we could use our data on Pulse, we can give actions over time to the patient, for example here a bleed. The other actions presented above are done in a similar way. For this, we have to change the *AnyAction* part of *VitalsMonitor.json* by adding the fields described in Listing **6**.

```
    "AnyAction": [{
  "PatientAction": {
    "Hemorrhage": {
      "Compartment":"RightLeg", // Right or Left
      "Type":"External",
      "Rate": { "ScalarVolumePerTime": { "Value":250.0, "Unit":"mL/min" } } // patient
    can endure a loss of 250mL/min
    }
  }
}]
```

Listing 6: Put a Hemorrhage in a scenario file

2.4.2 How to execute a scenario.

After implementing the desired scenario in a JSON file, it must be stored in the folder */install/bin/*, which is the location of the *PulseScenarioDriver* executable. The scenario can then be executed using the Linux command described in Listing **7**.

```
./PulseScenarioDriver VitalsMonitor.json
```

Listing 7: Execution command

The software then checks that the initialized patient is stable before starting the simulation. After confirmation of the stable state, the simulation displays in real time in the terminal what is happening.

```
...
[INFO] : [0(s)] Executing Scenario
[INFO] : [0(s)] [Action]
        Advance Time : 50(s)
[INFO] : [50.02(s)] [Action] 50.02(s), Environment Action : Change Environmental
    Conditions
        Environmental Conditions File: ./environments/Hypobaric4000m.json
[INFO] : [50.02(s)] [Action] 50.02(s), Patient Action : Consume Nutrients
        Charbohydrates: 415(g)
        Charbohydrates Digestion Rate: 0.5(g/min)
        Fat: 83(g)
        Fat Digestion Rate: 0.055(g/min)
        Protein: 99.6(g)
        Protein Digestion Rate: 0.071(g/min)
        Calcium: 1(g)
        Sodium: 1.5(g)
        Water: 3.7(L)
[INFO] : [50.02(s)] [Action]
        Advance Time : 60(min)
...
[INFO] : [60.74(s)] [Event] 60.74(s),  Patient has Hypoxia
...
[INFO] : [3650.04(s)] It took 486.788s to run this simulation
```

Listing 8: Display on the terminal during execution

The selected output data are stored in a CSV file generated by the simulation. The name of file is like the scenario file by adding "Results". For example, the results presented in Listing **9** are stored in *VitalsMonitorResults.csv.*

```
Time(s),HeartRate(1/min),OxygenSaturation,SkinTemperature(degC)
...
55.60,75.00,0.966,32.7
55.62,75.00,0.966,32.7
55.64,76.92,0.966,32.7
55.66,76.92,0.965,32.7
55.68,76.92,0.965,32.7
...
```

Listing 9: Results in CSV file

2.5 How does the cardiovascular system work?

The cardiovascular system is a large organ system comprised of the heart and the blood vessels. It serves as the body's primary transport and distribution system. The cardiovascular system is sometimes described as two separate circulations: the systemic circulation and the pulmonary circulation, which are connected. In the systemic circulation, oxygenated blood leaves the left side of the heart, travels through arteries and into the capillaries, and then returns as deoxygenated blood through the veins to the right side of the heart. From the right side of the heart, the deoxygenated blood travels to the pulmonary circulation through the pulmonary arteries, is re-oxygenated in the pulmonary capillaries, and then returns to the left side of the heart through the pulmonary veins. [**?**]

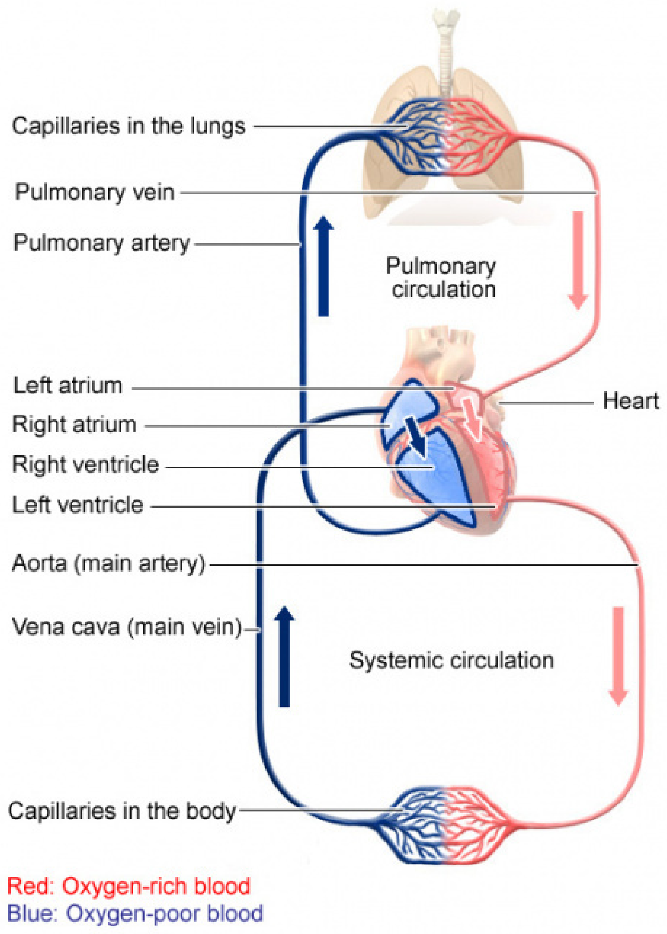

Figure 3: Diagram representing the cardiovascular system (Source :[?])

Whether the cardiovascular circulation is thought of as one closed circuit with a single dual-purpose pump or two interconnected circuits with separate synchronous pumps, the key combination of systemic and pulmonary vasculature serves as a distribution and exchange network, providing vital oxygen to the tissues and removing toxic carbon dioxide while distributing nutrients and other substances necessary for healthy physiologic function.

2.6 Cardiovascular modeling

The state of the cardiovascular system is determined at each time step by a three-step process: **Preprocessing**, **Processing** and **Postprocessing**. In the Preprocessing step, feedback from other systems, as well as internal system feedback, is processed to determine the system state. The Process step uses an already implemented function to calculate the new system state. The Post-processing step is used to prepare the system for the time advance. [5]

First, we have to initialize a patient with parameter values. So we have the initialization as well as the stabilization. The patient-specific homeostatic state of the cardiovascular circuit is computed. Next, all system parameters are initialized. The Cardiovascular System is then initialized to the resting state. Conditions are then applied by modifying system and patient parameters and restabilizing the engine. The available conditions in the cardiovascular system are anemia, heart failure, pericardial effusion, and pulmonary shunt.

- **Pre-processing** : Cardiac cycle calculations include methodology for updating the driving force (heart contraction and relaxation) of the cardiovascular system throughout the duration of a cardiovascular cycle (a single heart beat). This includes a set of systolic calculations that updates contractility at the beginning of the cycle to represent a heart contraction. For this we need the implemented function **HeartDriver** : [16]

```
void Cardiovascular::HeartDriver()
{
  // Reset start cardiac cycle event if it was activated by BeginCardiacCycle() last
     time step
  if (m_data.GetEvents().IsEventActive(eEvent::StartOfCardiacCycle))
    m_data.GetEvents().SetEvent(eEvent::StartOfCardiacCycle, false, m_data.
    GetSimulationTime());

  // m_StartSystole is set to true at the end of a cardiac cycle in order to setup
    the next cardiac cycle.
  // After the next cycle is prepared in BeginCardiacCycle, m_StartSystole is seet
    back to false.
  if (m_StartSystole)
    BeginCardiacCycle(); // we see it later.

  if (!m_data.GetEvents().IsEventActive(eEvent::CardiacArrest))
  {
    if (m_CurrentCardiacCycleTime_s >= m_CardiacCyclePeriod_s - m_data.GetTimeStep_s
    ())
      m_StartSystole = true; // A new cardiac cycle will begin next time step.

    AdjustVascularTone();
    CalculateHeartElastance();
  }
  // Compliance refers to the ability of an organ to adapt its volume to a pressure.
  m_pRightHeart->GetNextCompliance().SetValue(1.0 /
    m_RightHeartElastance_mmHg_Per_mL, VolumePerPressureUnit::mL_Per_mmHg);
  m_pLeftHeart->GetNextCompliance().SetValue(1.0 / m_LeftHeartElastance_mmHg_Per_mL,
     VolumePerPressureUnit::mL_Per_mmHg);

  // Now that the math is done we can increment the cardiac cycle time
  // Note that the cardiac cycle time (m_CurrentCardiacCycleTime_s) continues to
    increment until a cardiac cycle begins (a beat happens)
  // So for a normal sinus rhythm, the maximum cardiac cycle time is equal to the
    cardiac cycle period (m_CardiacCyclePeriod_s).
  // For any ineffective rhythm (no heart beat) the cardiac cycle time will be as
    long as it has been since the last time there was an effective beat.
  m_CurrentCardiacCycleTime_s += m_data.GetTimeStep_s();
}
```

Listing 10: The method HeartDriver(). Source : [16]

This function tracks the progress of the current cardiac cycle, and modifies the compliance of the left and right heart to drive the cardiovascular circuit. The reduced compliance at the beginning of the cycle acts to increase the pressure, driving flow out of the heart. The compliance is then reduced, allowing flow into the heart. This represents the systolic and diastolic portion of the cardiac cycle. The compliance is driven by an elastance equation. This function also keeps track of the cardiac cycle time and calls BeginCardiacCycle() at the start of systole portion of each cycle.

```
void Cardiovascular::BeginCardiacCycle()
{
```

```
m_data.GetEvents().SetEvent(eEvent::StartOfCardiacCycle, true, m_data.
  GetSimulationTime());

// Changes to the heart rate and other hemodynamic parameters are applied at the
  top of the cardiac cycle.
// Parameters cannot change during the cardiac cycle because the heart beat is
  modeled as a changing compliance.

double HeartDriverFrequency_Per_Min = m_data.GetCurrentPatient().
  GetHeartRateBaseline(FrequencyUnit::Per_min);
m_LeftHeartElastanceMax_mmHg_Per_mL = m_data.GetConfiguration().
  GetLeftHeartElastanceMaximum(PressurePerVolumeUnit::mmHg_Per_mL);
m_RightHeartElastanceMax_mmHg_Per_mL = m_data.GetConfiguration().
  GetRightHeartElastanceMaximum(PressurePerVolumeUnit::mmHg_Per_mL);

// Apply baroreceptor reflex effects
/// \todo need to reset the heart elastance min and max at the end of each
  stabiliation period in AtSteadyState()
if (m_data.GetNervous().GetBaroreceptorFeedback() == eSwitch::On)
{
  m_LeftHeartElastanceMax_mmHg_Per_mL *= m_data.GetNervous().
  GetBaroreceptorHeartElastanceScale().GetValue();
  m_RightHeartElastanceMax_mmHg_Per_mL *= m_data.GetNervous().
  GetBaroreceptorHeartElastanceScale().GetValue();
  HeartDriverFrequency_Per_Min *= m_data.GetNervous().
  GetBaroreceptorHeartRateScale().GetValue();
}
if (m_data.GetNervous().GetChemoreceptorFeedback() == eSwitch::On)
{
  // Chemoreceptor and drug effects are deltas rather than multipliers, so they
  are added.
  HeartDriverFrequency_Per_Min += m_data.GetNervous().
  GetChemoreceptorHeartRateScale().GetValue();
}

// Apply drug effects
if (m_data.GetDrugs().HasHeartRateChange())
  HeartDriverFrequency_Per_Min += m_data.GetDrugs().GetHeartRateChange(
  FrequencyUnit::Per_min);
BLIM(HeartDriverFrequency_Per_Min, m_data.GetCurrentPatient().GetHeartRateMinimum(
  FrequencyUnit::Per_min), m_data.GetCurrentPatient().GetHeartRateMaximum(
  FrequencyUnit::Per_min));

//Apply heart failure effects
m_LeftHeartElastanceMax_mmHg_Per_mL *= m_LeftHeartElastanceModifier;

// Now set the cardiac cycle period and the cardiac arrest event if applicable
if (m_EnterCardiacArrest)
{
  m_data.GetEvents().SetEvent(eEvent::CardiacArrest, true, m_data.
  GetSimulationTime());
  m_CardiacCyclePeriod_s = 1.0e9; // Not beating, so set the period to a large
  number (1.0e9 sec = 31.7 years)
  RecordAndResetCardiacCycle();
  GetHeartRate().SetValue(0.0, FrequencyUnit::Per_min);
}
else
{
  if (HeartDriverFrequency_Per_Min == 0)
  {
    m_CardiacCyclePeriod_s = 5.0; // Can't divide by 0, but we want to check this
  again in a while to see if we can get out of asystole
    GetHeartRate().SetValue(0.0, FrequencyUnit::Per_min); // Will put patient into
   asystole
  }
```

```
    else
    {
      m_CardiacCyclePeriod_s = 60.0 / HeartDriverFrequency_Per_Min;
    }
  }

  // Reset the systole flag and the cardiac cycle time
  m_StartSystole = false;
  m_CurrentCardiacCycleTime_s = 0.0;
}
```

Listing 11: The method BeginCardiacCycle(). Source : [16]

This function is directed **HeartDriver()**. It sets up the evolution of the current cardiac cycle. It is used to apply the effects of drugs or exercise on the cardiovascular system. These effects will persist until the end of the cardiac cycle, and this function will then be called again if there is no cardiac arrest. Modifications to heart rate and heart compliance are calculated by **BeginCardiacCycle()** and applied for the remained of the current cardiac cycle. Changes to things like heart rate and heart contractility can only occur at the top of the current cardiac cycle, after the last cardiac cycle has completed. This helps to avoid discontinuous behavior, such as the complete cessation of heart function mid-contraction.

In addition, in this **pre-process**, we have the **CalculateHeartElastance()** method. This method tracks the progression of the current cardiac cycle and changes the compliance of the left and right heart to drive the cardiovascular circuit. Reduced compliance at the beginning of the cycle results in increased pressure, causing flow to leave the heart. Compliance is then reduced, allowing flow to enter the heart. The reduction and increase in compliance represent the systolic and diastolic parts of the cardiac cycle respectively.

```
void Cardiovascular::CalculateHeartElastance()
{
  //Shape parameters, used to define double hill functional form of the elastance
  double alpha1 = 0.303;
  double alpha2 = 0.508;
  double n1 = 1.32;
  double n2 = 21.9;
  double maxShape = 0.598;
  double oxygenDeficitEffect = 1.0;

  if (m_data.GetEvents().IsEventActive(eEvent::MyocardiumOxygenDeficit) == true)
  {
    double eventDuration = m_data.GetEvents().GetEventDuration(eEvent::
    MyocardiumOxygenDeficit, TimeUnit::s);
    oxygenDeficitEffect = pow(-3E-9*eventDuration, 2) + 8E-6*eventDuration + 0.9865;
  }

  double normalizedCardiacTime = m_CurrentCardiacCycleTime_s /
    m_CardiacCyclePeriod_s;
  double elastanceShapeFunction = (pow(normalizedCardiacTime / alpha1, n1) / (1.0 +
    pow(normalizedCardiacTime / alpha1, n1)))*(1.0 / (1.0 + pow(
    normalizedCardiacTime / alpha2, n2))) / maxShape;

  m_LeftHeartElastance_mmHg_Per_mL = oxygenDeficitEffect * ((
    m_LeftHeartElastanceMax_mmHg_Per_mL - m_LeftHeartElastanceMin_mmHg_Per_mL)*
    elastanceShapeFunction + m_LeftHeartElastanceMin_mmHg_Per_mL);
  m_RightHeartElastance_mmHg_Per_mL = oxygenDeficitEffect * ((
    m_RightHeartElastanceMax_mmHg_Per_mL - m_RightHeartElastanceMin_mmHg_Per_mL)*
    elastanceShapeFunction + m_RightHeartElastanceMin_mmHg_Per_mL);
}
```

Listing 12: The method CalculateHeartElastance().

Finally, the **ProcessActions()** method modifies the cardiovascular parameters and the circuit properties due to actions specified by the user. The actions found in the Actions process are: cardiopulmonary resuscitation, hemorrhage, pericardial effusion and cardiac arrest.

```cpp
void Cardiovascular::ProcessActions()
{
  TraumaticBrainInjury();
  Hemorrhage();
  PericardialEffusion();
  CPR();
  CardiacArrest();
}
```

Listing 13: The method ProcessActions().

So we get our method for the **pre-process** :

```cpp
void Cardiovascular::PreProcess()
{
  // Locate the cardiac cycle in time (systole, diastole)
  // and do the appropriate calculations based on the time location.
  HeartDriver();
  ProcessActions();
  UpdateHeartRhythm();
  // The heart rhythm is set to either Asystole (Asystole is defined as a cessation
    of the heart beat.) or NormalSinus based on if the patient has an active cardiac
     arrest or has triggered the asystole event some other way.

}
```

Listing 14: The method Preprocess().

- **Process** : The function already implemented **CalculateVitalSigns()** takes the current time step's circuit quantities to calculate important system-level quantities for the current time step. The system pressures and flow rates related to shunting are calculated here. In addition, different cardiac events are triggered. [16]

```cpp
void Cardiovascular::CalculateVitalSigns()
{
  // Grab data from the circuit in order to calculate a running mean
  const double AortaNodePressure_mmHg = m_Aorta->GetPressure(PressureUnit::mmHg);
  const double AortaNodeCO2PartialPressure_mmHg = m_AortaCO2 == nullptr ? 0 :
    m_AortaCO2->GetPartialPressure(PressureUnit::mmHg); // This is here so we can
    Tune circuit w/o substances
  const double LeftPulmonaryArteryVolume_mL = m_LeftPulmonaryArteries->GetVolume(
    VolumeUnit::mL);
  const double RightPulmonaryArteryVolume_mL = m_RightPulmonaryArteries->GetVolume(
    VolumeUnit::mL);
  const double TotalPulmonaryArteryVolume_mL = LeftPulmonaryArteryVolume_mL +
    RightPulmonaryArteryVolume_mL;
  const double LeftPulmonaryArteryPressure_mmHg = m_LeftPulmonaryArteries->
    GetPressure(PressureUnit::mmHg);
    ...
    ...

  // Record high and low values to compute for systolic and diastolic pressures:
  if (AortaNodePressure_mmHg > m_CardiacCycleAortaPressureHigh_mmHg)
    m_CardiacCycleAortaPressureHigh_mmHg = AortaNodePressure_mmHg;
  if (AortaNodePressure_mmHg < m_CardiacCycleAortaPressureLow_mmHg)
    m_CardiacCycleAortaPressureLow_mmHg = AortaNodePressure_mmHg;
  if (PulmonaryArteryNodePressure_mmHg >
    m_CardiacCyclePulmonaryArteryPressureHigh_mmHg)
    m_CardiacCyclePulmonaryArteryPressureHigh_mmHg =
    PulmonaryArteryNodePressure_mmHg;
```

```cpp
if (PulmonaryArteryNodePressure_mmHg < m_CardiacCyclePulmonaryArteryPressureLow_mmHg)
  m_CardiacCyclePulmonaryArteryPressureLow_mmHg = PulmonaryArteryNodePressure_mmHg;

// Get Max of Left Ventricle Volume over the course of a heart beat for end diastolic volume
if (LHeartVolume_mL > m_CardiacCycleDiastolicVolume_mL)
  m_CardiacCycleDiastolicVolume_mL = LHeartVolume_mL;

/// \todo Make sure irreversible state is hit before we get here.
if (m_CardiacCycleAortaPressureLow_mmHg < -2.0)
{
  Fatal("Diastolic pressure has fallen below zero.");
  /// \error Fatal: Diastolic pressure has fallen below -2
}
if (m_CardiacCycleAortaPressureHigh_mmHg > 700.0)
{
  Fatal("Systolic pressure has exceeded physiologic range.");
  /// \error Fatal: Systolic pressure has exceeded 700
}

// Pressures\Flows from circuit
GetArterialPressure().SetValue(AortaNodePressure_mmHg, PressureUnit::mmHg);
GetPulmonaryArterialPressure().SetValue(PulmonaryArteryNodePressure_mmHg, PressureUnit::mmHg);
GetCentralVenousPressure().SetValue(VenaCavaPressure_mmHg, PressureUnit::mmHg);
GetCerebralBloodFlow().Set(m_Brain->GetInFlow());
GetIntracranialPressure().Set(m_Brain->GetPressure());
GetCerebralPerfusionPressure().SetValue(GetMeanArterialPressure(PressureUnit::mmHg) - GetIntracranialPressure(PressureUnit::mmHg), PressureUnit::mmHg);

if (m_data.GetState() > EngineState::AtSecondaryStableState)
{// Don't throw events if we are initializing

  // Check for different events :


// Check for hypovolemic shock
/// \event Patient: Hypovolemic Shock: blood volume below 65% of its normal value
  if (GetBloodVolume().GetValue(VolumeUnit::mL) <= (m_data.GetConfiguration().GetMinimumBloodVolumeFraction()* m_data.GetCurrentPatient().GetBloodVolumeBaseline(VolumeUnit::mL)))
  {
    m_data.GetEvents().SetEvent(eEvent::HypovolemicShock, true, m_data.GetSimulationTime());
  }
  else
  {
    m_data.GetEvents().SetEvent(eEvent::HypovolemicShock, false, m_data.GetSimulationTime());
  }

  if (GetMeanArterialPressure().GetValue(PressureUnit::mmHg) <= 20)
  {
      m_data.GetEvents().SetEvent(eEvent::CardiovascularCollapse, true, m_data.GetSimulationTime());
  }
  else
  {
      m_data.GetEvents().SetEvent(eEvent::CardiovascularCollapse, false, m_data.GetSimulationTime());
  }

  //Check for cardiogenic shock
```

```
    if (GetCardiacIndex().GetValue(VolumePerTimeAreaUnit::L_Per_min_m2) < 2.2 &&
      GetSystolicArterialPressure(PressureUnit::mmHg) < 90.0 &&
      GetPulmonaryCapillariesWedgePressure(PressureUnit::mmHg) > 15.0)
    {
      /// \event Patient: Cardiogenic Shock: Cardiac Index has fallen below 2.2 L/
    min-m^2, Systolic Arterial Pressure is below 90 mmHg, and Pulmonary Capillary
    Wedge Pressure is above 15.0.
      ///
      m_data.GetEvents().SetEvent(eEvent::CardiogenicShock, true, m_data.
    GetSimulationTime());
    }
    else
    {
      m_data.GetEvents().SetEvent(eEvent::CardiogenicShock, false, m_data.
    GetSimulationTime());
    }

    //Check for Tachycardia, Bradycardia, and asystole
    /// \event Patient: Tachycardia: heart rate exceeds 100 beats per minute.  This
    state is alleviated if it decreases below 90.
    if (GetHeartRate().GetValue(FrequencyUnit::Per_min) < 90)
      m_data.GetEvents().SetEvent(eEvent::Tachycardia, false, m_data.
    GetSimulationTime());
    if (GetHeartRate().GetValue(FrequencyUnit::Per_min) > 100)
      m_data.GetEvents().SetEvent(eEvent::Tachycardia, true, m_data.
    GetSimulationTime());
    /// \event Patient: Bradycardia: heart rate falls below 60 beats per minute.
    This state is alleviated if it increases above 65.
    if (GetHeartRate().GetValue(FrequencyUnit::Per_min) < 60)
      m_data.GetEvents().SetEvent(eEvent::Bradycardia, true, m_data.
    GetSimulationTime());
    if (GetHeartRate().GetValue(FrequencyUnit::Per_min) > 65)
      m_data.GetEvents().SetEvent(eEvent::Bradycardia, false, m_data.
    GetSimulationTime());
    if (GetHeartRate().GetValue(FrequencyUnit::Per_min) == 0 || m_data.GetActions().
    GetPatientActions().HasCardiacArrest())
    {
      m_data.GetEvents().SetEvent(eEvent::Asystole, true, m_data.GetSimulationTime()
    );
    }
    else
    {
      m_data.GetEvents().SetEvent(eEvent::Asystole, false, m_data.GetSimulationTime
    ());
    }
  }

  // Irreversible state if asystole persists.
  if (GetHeartRhythm() == eHeartRhythm::Asystolic)
  {
    /// \event Patient: Irreversible State: heart has been in asystole for over 45
    min:
    if (m_data.GetEvents().GetEventDuration(eEvent::Asystole, TimeUnit::s) > 2700.0)
     //
    {
      /// \irreversible Heart has been in asystole for over 45 min
      m_data.GetEvents().SetEvent(eEvent::IrreversibleState, true, m_data.
    GetSimulationTime());
      m_ss << "Asystole has occurred for " << m_data.GetEvents().GetEventDuration(
    eEvent::Asystole, TimeUnit::s) << " seconds, patient is in irreversible state.";
      Fatal(m_ss);
    }
  }

  // Compute blood volume
  double blood_mL = 0;
  for (SELiquidCompartment* cmpt : m_data.GetCompartments().
    GetVascularLeafCompartments())
  {
    if (cmpt->HasVolume() && cmpt != m_Pericardium) //Don't include pericardium
    {
      blood_mL += cmpt->GetVolume(VolumeUnit::mL);
    }
  }
  GetBloodVolume().SetValue(blood_mL, VolumeUnit::mL);
}
```
Listing 15: The method CalculateVitalSigns(). Source : [16]

- **Post-processing** : The Post-process step moves everything calculated in **Process** from the next time step calculation to the current time step calculation. This allows all other systems access to the information when completing their Preprocess analysis for the next time step. [16]

2.7 The Cardiovascular Circuit

The cardiovascular circuit estimates blood pressure, flow, and volume for organs that are represented by several compartments. These compartments are comprised of lumped parameter models that use resistors and capacitors. The system is discretized into nodes that are connected by paths The circuit used to represent the cardiovascular system was designed to provide a level of resolution and fidelity that meets the requirements of different projects. For example, to provide a means for clearing drugs and substances from the bloodstream, the liver and kidneys must have blood flow, pressure, and volume calculations.

The nodes serve as connection points for the paths and are the places where the pressures are measured. Each node contains a pressure value, which is given relative to the atmospheric reference node (indicated in the diagram by the equipotential symbol). The paths contain information about the flow rate (volume per time). The document Circuit Methodology contains more information on circuit definitions and modelling. The Substance Transport Methodology document contains more information on substance transport. In general, nodes contain "cross-sectional" information and paths contain "cross-sectional" information. The CV circuit elements are used to model the fluid dynamics of the human CV system (hemodynamics).hemodynamic pressure and flow are calculated from the lumped parameters determined by the circuit element. (using equations already implemented and solved by the circuit solver) [5]

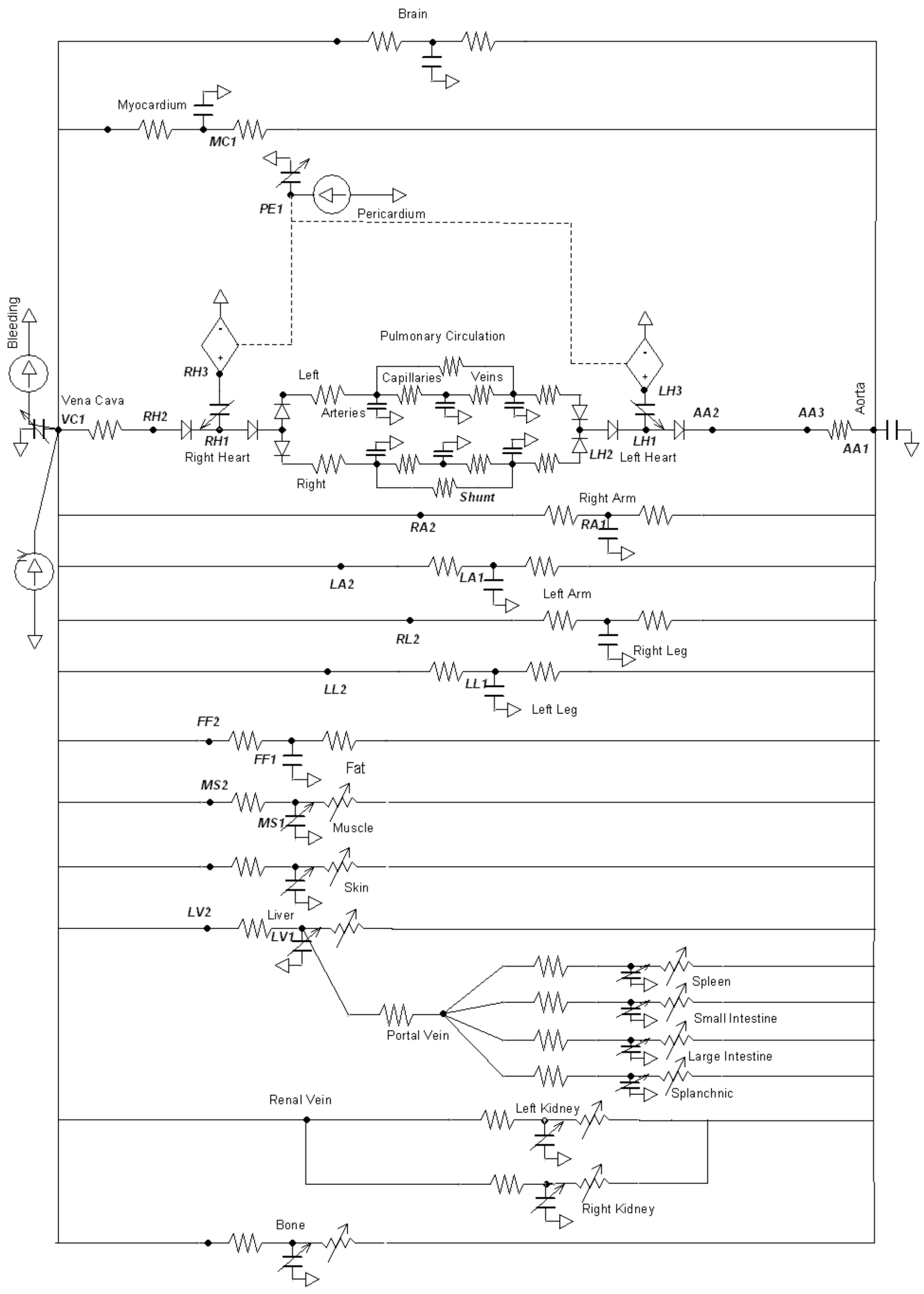

Figure 4: The cardiovascular circuit (Source : Pulse [5])

3 Another approach : Biofluid modelling of the coupled eye-brain system.

3.1 Context

We will look at another modelling approach, that of an eye-brain coupling, which may be similar to that of Pulse. Microgravity conditions have been observed to induce structural and functional changes in the eyes of many astronauts, posing serious problems for both the astronauts and their missions in space. This syndrome, also known as Spaceflight Associated Neuro-ocular Syndrome (SANS), is characterized by many apparently unrelated and often non-concurrent symptoms such as choroidal folds. The current understanding of how weightlessness environment affects the human body and may lead to SANS development is still quite rudimentary. Various studies of the symptoms experienced by astronauts during long-term missions (four to six months) have been performed, but their validity is hampered by the small size of the subjects' cohort. To overcome this difficulty, ground-based microgravity laboratory models have been proposed, the most significant of which is the long head down tilt experimental procedure that is used to simulate the effects of microgravity on the **cardiovascular system**. The idea in our case is not to model microgravity, but to understand the patterns resulting from this modelling. [17]

A complementary approach to experimental studies is the use of mathematical modeling as a tool to investigate theoretically the role of various factors potentially contributing to SANS and help elucidate the mechanisms of their interactions and their implications in structural and functional changes in the eye.

3.2 The model.

It is a lumped parameter model of the brain and eyes, connected with a highly simplified model of the body, the electrical analogue of which is shown in the schema below The aim of the model, which describes pressure and flow distributions in the brain and eyes, is to predict fluid redistribution in the upper body vasculature and variations of the intraocular pressure and the intracranial pressure following exposure to a microgravity environment. Indeed, studies suggest that two main mechanisms may be involved in SANS pathophysiology :

- changes in the vascular system and fluid distribution.
- changes in the brain/central nervous system and intracranial pressure.

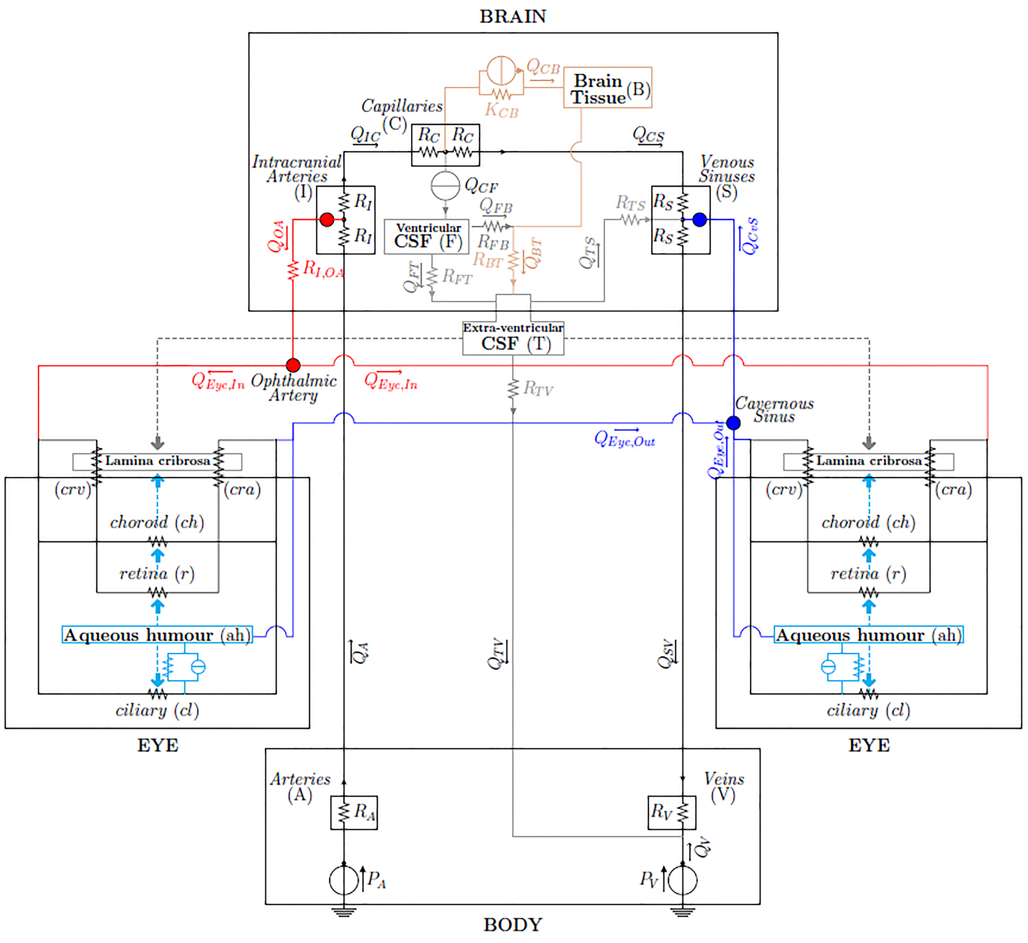

Figure 5: Model of fluid flows in the brain and eyes (Source : [17])

The physiological system is subdivided into a number of linked, interacting compartments, each of which contains a single physical constituent, such as blood, cerebrospinal fluid, interstitial fluid and aqueous humor. The brain and eye models are coupled to each other and are linked to the rest of the body via a highly simplified model consisting of two compartments: central arteries and central veins. This description corresponds to a model for the human physiology of the upper part of the body. [17]

References

[1] What is Pulse? (2017-2021). *Pulse Physiology Engine.* https://pulse.kitware.com/index.html

[2] Common Data Model and Software Framework. (2017-2021). *Pulse Physiology Engine.* https://pulse.kitware.com/_c_d_m.html

[3] CDM Tables. (2017-2021). *Pulse Physiology Engine.* https://pulse.kitware.com/_c_d_m_tables.html

[4] Scenario Files. (2017-2021). *Pulse Physiology Engine.* https://pulse.kitware.com/_scenario_file.html

[5] Cardiovascular Methodology. (2017-2021). *Pulse Physiology Engine.* https://pulse.kitware.com/_cardiovascular_methodology.html

[6] Circuit Methodology. (2017-2021). *Pulse Physiology Engine.* https://pulse.kitware.com/_circuit_methodology.html

[7] Patient Methodology. (2017-2021). *Pulse Physiology Engine.* https://pulse.kitware.com/_patient_methodology.html

[8] Qui sommes-nous. *Groupama-FDJ Equipe Cycliste.* https://www.equipecycliste-groupama-fdj.fr/qui-sommes-nous/

[9] Fat. *Wikipedia.* https://en.wikipedia.org/wiki/Fat

[10] Protein. *Wikipedia.* https://en.wikipedia.org/wiki/Protein

[11] Calcium. *Wikipedia.* https://en.wikipedia.org/wiki/Calcium

[12] Anatomy and Physiology : Cardiac Output. (2021). *Anatomy and Physiology.* http://www.anaphy.com/cardiac-output/#:~:text=In%20a%20healthy%20adult%20with%2072%20bpm%2C%20cardiac,which%20heart%20pumps%20blood%20and%20its%20maximum%20capacity.

[13] Vinik, A-I., Erbas, T. (2013, November 11). Diabetic autonomic neuropathy. *Handbook of Clinical Neurology, Volume 117.* 279-294. https://www.sciencedirect.com/science/article/pii/B9780444534910000225

[14] Total Metabolic Rate. (2009, July 23). *Wisdom That Heals : A Blog about Health, Nutrition and Natural Healing.* https://wisdomthatheals.wordpress.com/tag/total-metabolic-rate/

[15] Partial pressure. *Wikipedia.* https://en.wikipedia.org/wiki/Partial_pressure

[16] GitLab *SECardiovascularSystem.cpp* https://gitlab.kitware.com/physiology/engine/-/blob/master/src/cpp/cdm/system/physiology/SECardiovascularSystem.cpp

[17] PlosOne *Biofluid modeling of the coupled eye-brain system and insights into simulated microgravity conditions* https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0216012p