Med Uni Graz *Pioneering Minds*

FORCEPSS

A framework for Cardiac Electrophysiology Simulations Standardization

Matthias Gsell, Aurel Neic, Martin Bishop,
Karli Gillette, Anton Prassl, Christoph Augustin,
Edward Vigmond, Gernot Plank

matthias.gsell@medunigraz.at
https://ccl.medunigraz.at

OpenCARP User Meeting, 15.07.2024

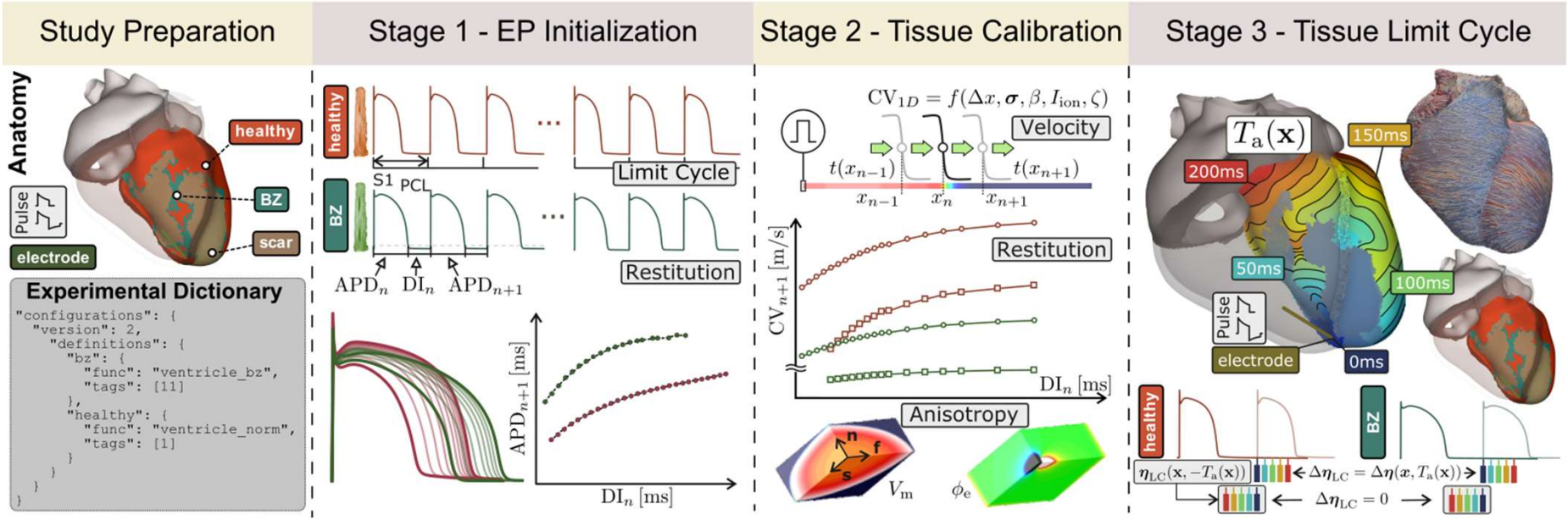
FFG Forschung wirkt.
InstaTwin
FO999891133

Background & Motivation

- Computer simulation of cardiac electrophysiology (**CEP**) is an important research tool.
- Simulations are increasingly being used in clinical and industrial applications.
- Typical workflow of a **CEP** simulation starts with the anatomical model building and the initialization & calibration of the **EP**.
- Initialization & calibration steps are common and generalizable.
- Most **CEP** studies re-implement these steps which lacks reproducibility.
- ***ForCEPSS*** was integrated in the Python-based ***carputils*** framework, utilizing the ***openCARP*** simulator.
- ***ForCEPSS*** provides a robust yet flexible framework to standardize and automate **CEP** studies.

Background & Motivation

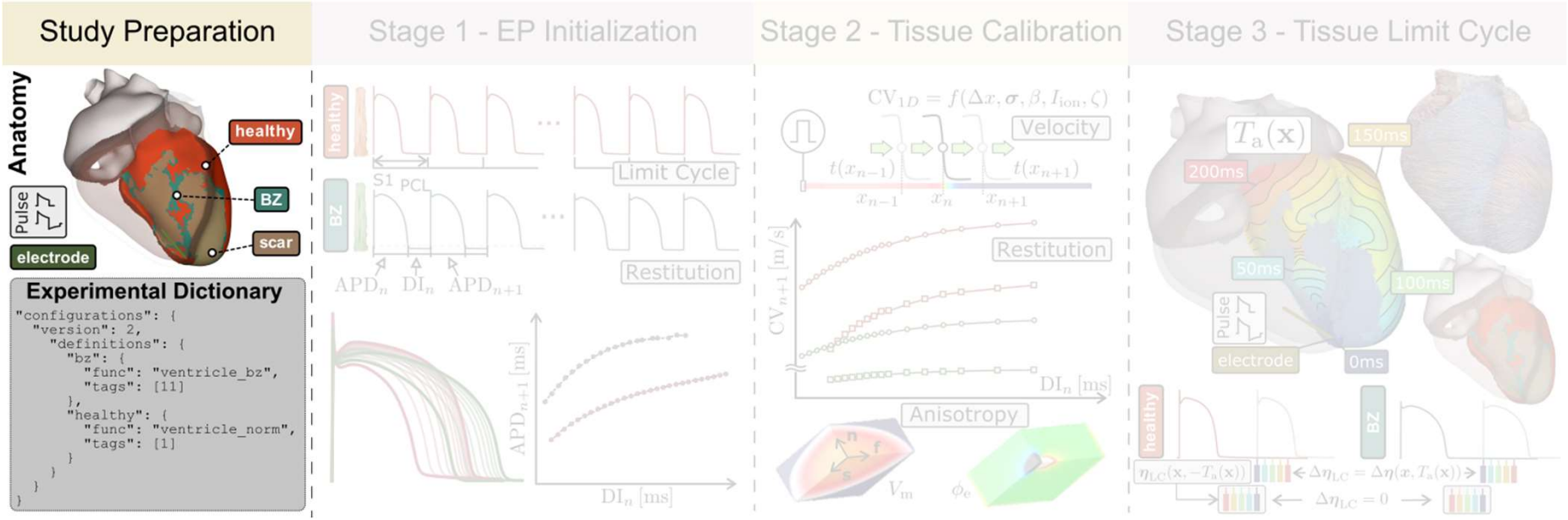
Standardize workflow for **EP** simulations:



Cell Level — Tissue Level — Organ Level

Background & Motivation

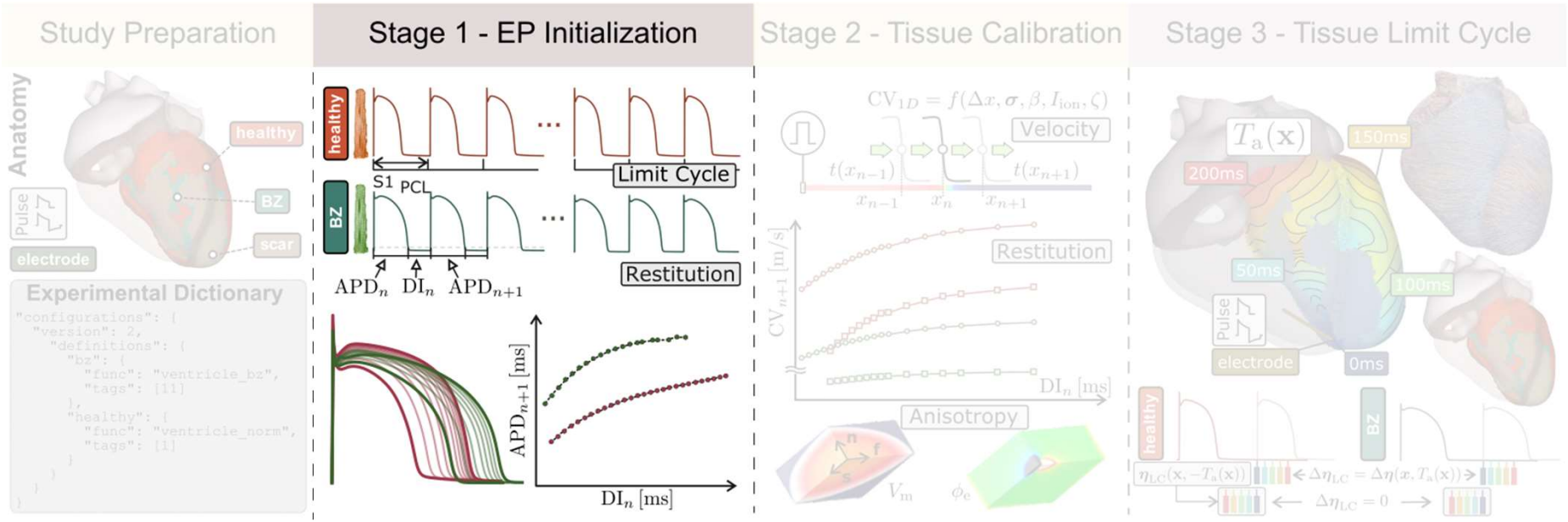
Standardize workflow for **EP** simulations:



Study Preparation: Define simulation related components such as anatomy, functional regions and their electrophysiological properties.

Background & Motivation

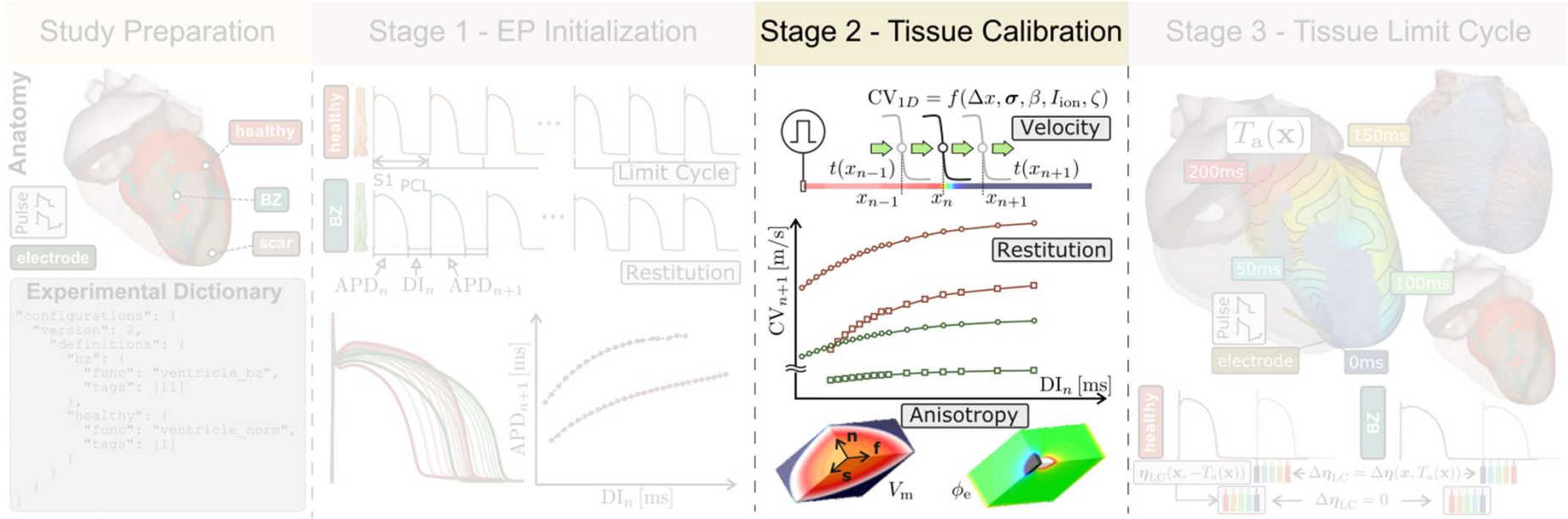
Standardize workflow for **EP** simulations:



EP Initialization: Compute stable limit cycle of all cell models in the simulation for a given cycle length of interest. Determine action potential duration (**APD**) restitution properties and tune these accordingly.

Background & Motivation

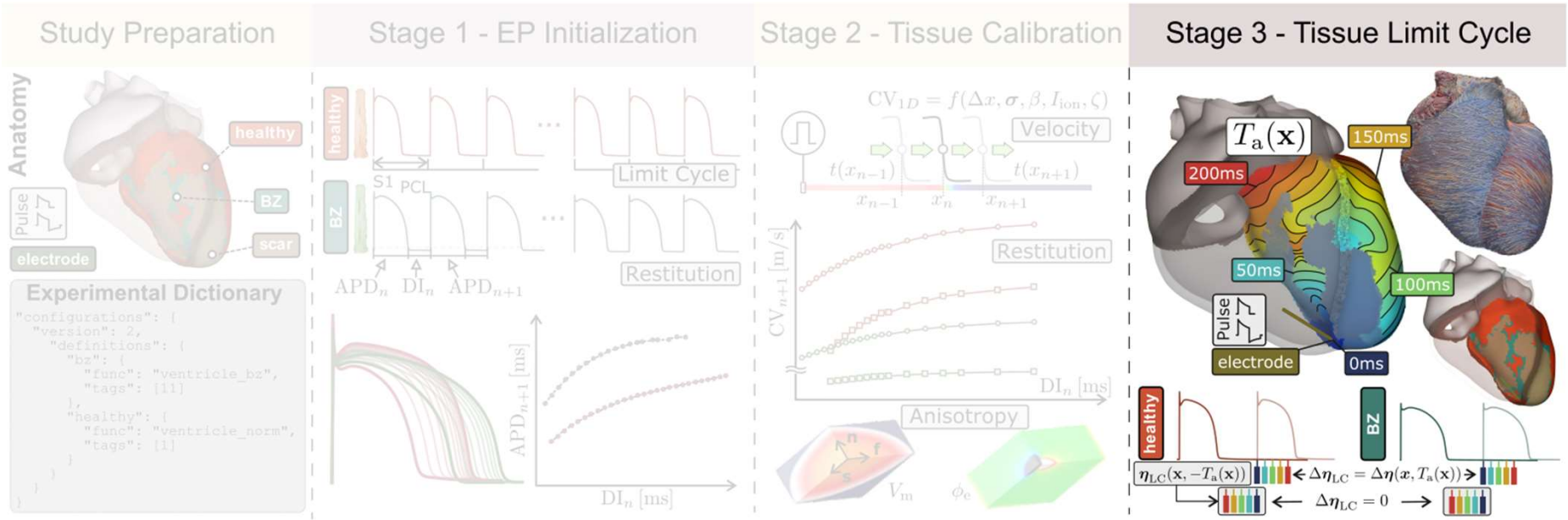
Standardize workflow for **EP** simulations:



Tissue Calibration: Define conduction velocities (**CV**s) and conductivities. Determine **CV** restitution properties for given mesh resolution and numerical settings.

Background & Motivation

Standardize workflow for **EP** simulations:



Tissue Limit Cycle: Define tissue anatomy, electrodes (location, size, shape), pacing protocols.

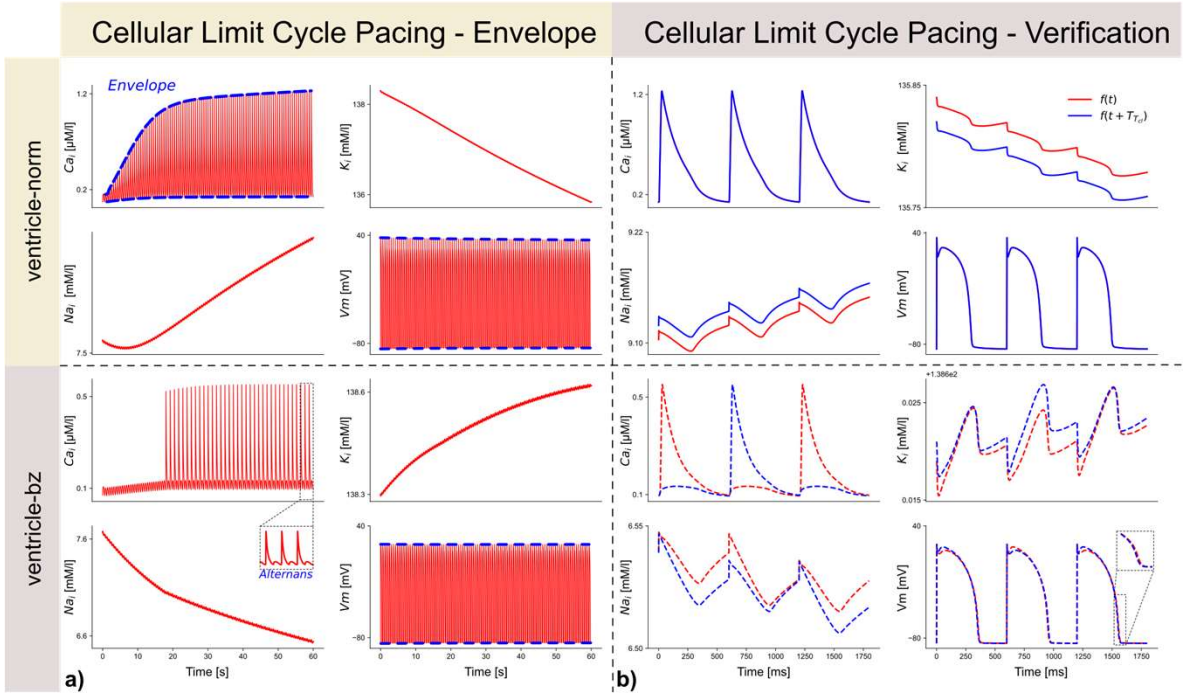
Stage 1: EP Initialization

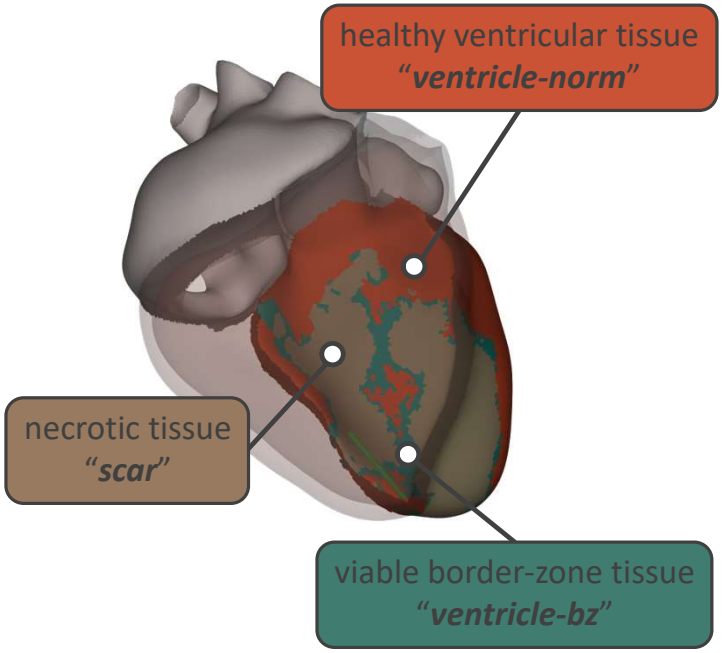
Aim is to find a stable limit cycle of the cellular **EP** for a given pacing cycle length (**PCL**) $T_{cl} > 0$.

The state vector $\boldsymbol{\eta}$ of each phenotype p satisfies the following condition:

$$\boldsymbol{\eta}_p(t) = \boldsymbol{\eta}_p(t + T_{cl})$$

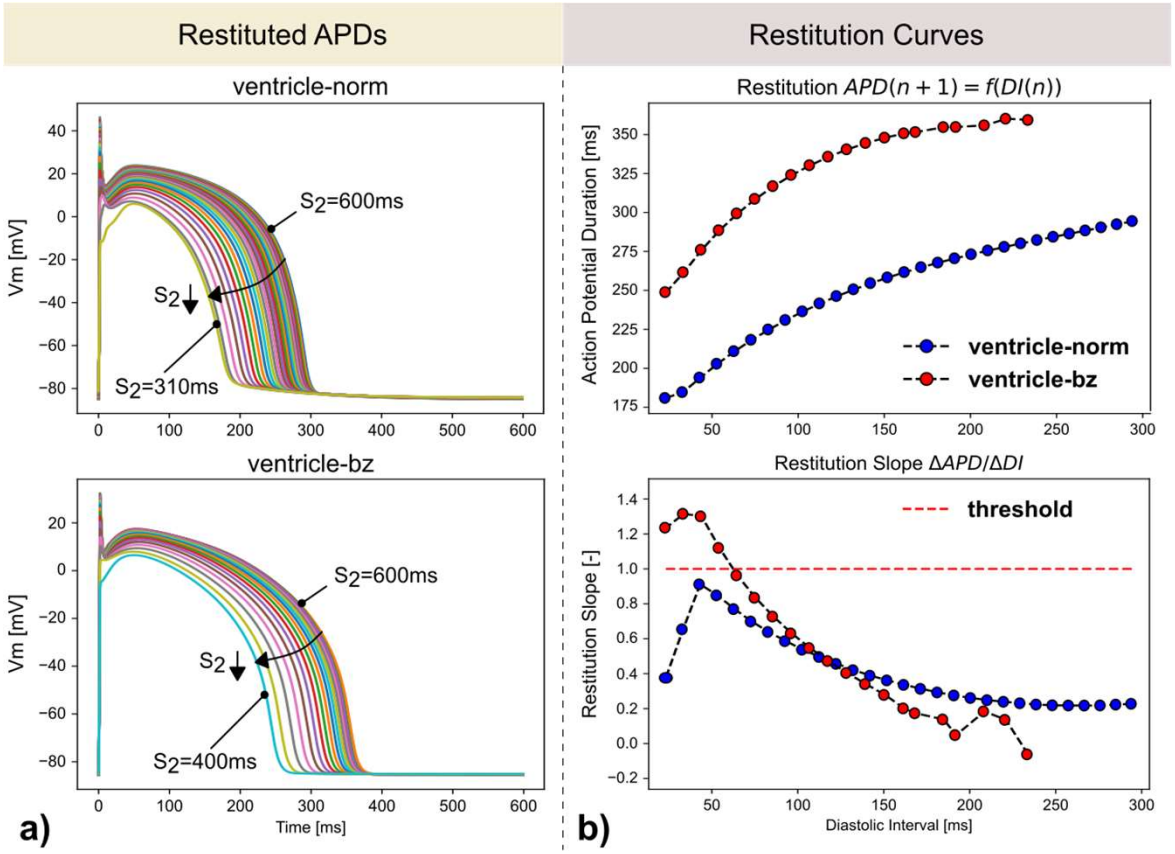
for $t \geq T_{stab} > T_{cl}$.

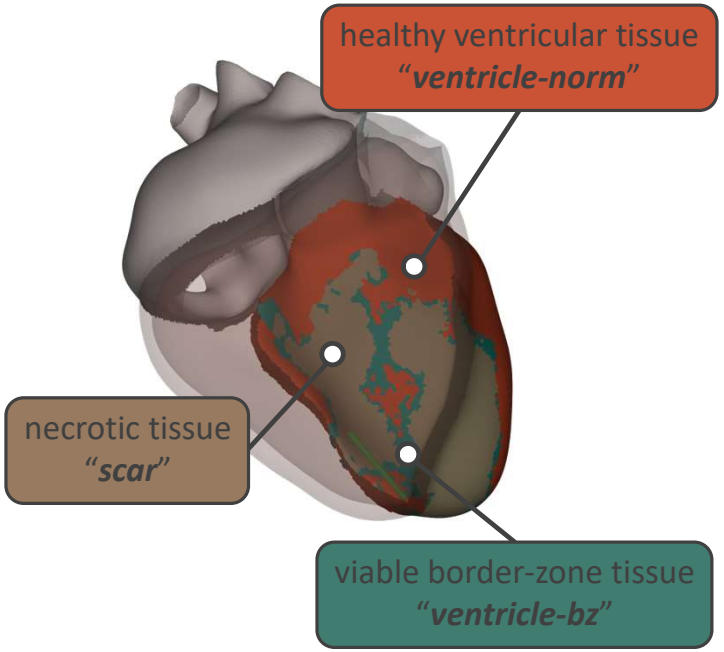




Stage 1: EP Initialization

APD restitution experiments can be performed.





Stage 1: EP Initialization


```
"functions": {
     "version": 2,
     "calibration": {
          "tuning_dir": "tune"
     },
     "definitions": {
          "ventricle-norm": {
               "ionic_model": {
                    "model": "tenTusscherPanfilov",
                    "model_par": null,
                    "initialization": {
                         "bcl": 600.0,
                         "num_cycles": 100
                    }
               },
               "conductivity": {},
               "conduction_velocity": {
                    "reference": {}
               }
          },
          "ventricle-bz": { ... },
          "scar": {
               "ionic_model": null
          }
     }
}
```

- *ep_init.py --plan plan.json --list-funcs*
  list all **EP** functions defined in the plan file
- *ep_init.py --plan plan.json --visualize --cycles 200 --bcl 550*
  run ***bench.opt*** to initialize all cell-models (no automatic limit cycle detection) and visualize with ***limpetgui***
- *ep_init.py --plan plan.json --verify*
  manual check whether a limit cycle has been reached
- *ep_init.py --plan plan.json --update --cycles 200 --bcl 550*
  update the initialization sections in the plan file

Stage 1: EP Initialization

```
"protocols": {
     "version": 2,
     "apd_restitution": {
          "protocol0": {
               "type": "S1S2",
               "bcl": 600.0,
               "CI0": 50.0,
               "CI1": 600.0,
               "pp_beats": 20,
               "pm_beats": 5,
               "pm_dec": 10,
          },
          "protocol1": {
               "type": “dynamic",
               "bcl": 600.0,
               "CI0": 50.0,
               "CI1": 600.0,
               "pp_beats": 20,
               "pm_beats": 5,
               "pm_dec": 10,
          },
           ...
     }
}
```

- *ep_init.py --plan plan.json --list-funcs*
  list all **EP** functions defined in the plan file
- *ep_init.py --plan plan.json --visualize --cycles 200 --bcl 550*
  run ***bench.opt*** to initialize all cell-models (no automatic limit cycle detection) and visualize with ***limpetgui***
- *ep_init.py --plan plan.json --verify*
  manual check whether a limit cycle has been reached
- *ep_init.py --plan plan.json --update --cycles 200 --bcl 550*
  update the initialization sections in the plan file
- *ep_init.py --plan plan.json --restitute --plot-apd-restitution --apd-restitution-protocol protocol0*
  run and plot **APD** restitution curve

https://opencarp.org/documentation/examples/01_ep_single_cell/02b_apd_restitution

Stage 2: Tissue Calibration

In the second stage for each region, tissue conductivities are adjusted to match prescribed orthotropic **CV**s.

- All factors such as mesh resolution, myocyte geometry and numerical settings that have an influence on the **CV** are considered.

- If the bidomain equivalent monodomain conductivity tensor

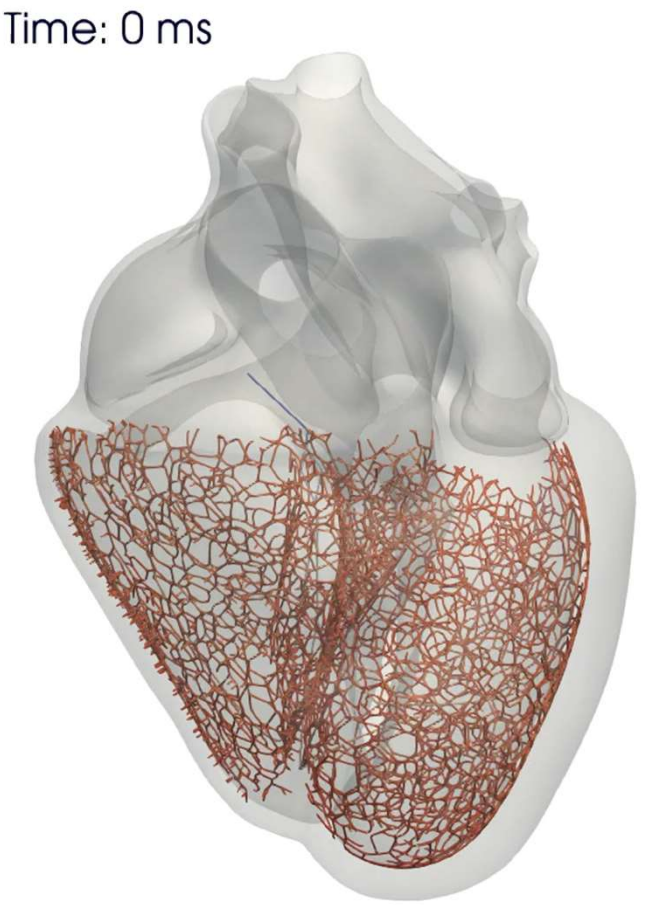
$$\sigma_m = \sigma_i \cdot (\sigma_i + \sigma_e)^{-1} \cdot \sigma_e$$

is used, σ_i and σ_e can be tuned by keeping their ratio constant or only σ_i can be tuned keeping σ_e constant.

- If the intracellular conductivity tensor is preferred, i.e.

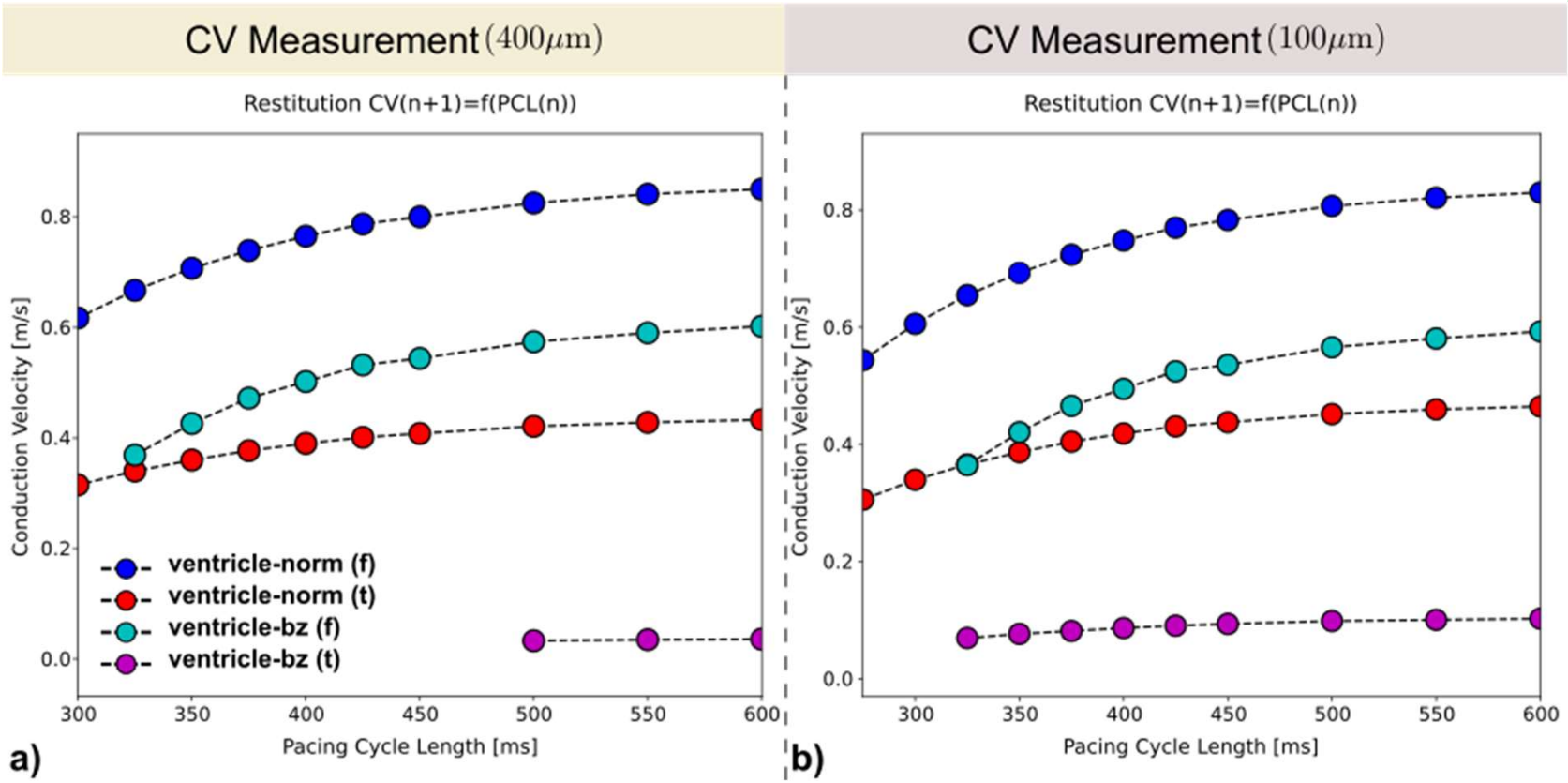
$$\sigma_m = \sigma_i,$$

then only σ_i can be tuned.



Stage 2: Tissue Calibration

CV restitution experiments can be performed



Stage 2: Tissue Calibration


```
"functions": {
     "version": 2,
     "calibration": {
          "tuning_dir": "tune"
     },
     "definitions": {
          "ventricle-norm": {
               "ionic_model": { ... },
               "conductivity": {
                    "gil": 0.255,
                    "gel": 0.625,
                    "git": 0.0775,
                    "get": 0.236,
                    "gin": 0.0775,
                    "gen": 0.236,
                    "surf2vol": 0.14
               },
               "conduction_velocity": {
                    "reference": {
                         "vf": 0.8275,
                         "vs": 0.4591,
                         "vn": 0.4591
                    }
               }
          },
           ...
     }
}
```

Stage 2: Tissue Calibration


```
"functions": {
     "version": 2,
     "calibration": {
          "tuning_dir": "tune"
     },
     "definitions": {
          "ventricle-norm": { ... },
          "ventricle-bz": { ... },
          "scar": { ... }
     }
},
"solver_setup": {
     "version": 2,
     "source_model": "monodomain",
     "beqm": true,
     "fixge": true,
     "dt": 25.0,
     "dx": 250.0,
     "dtsubstep": 1,
     "lumping": false,
     "opsplit": true,
     "tolerance": 1.0e.6,
     "ts": 1,
}
```

- *tissue_calibrate.py --plan plan.json --resolution 400 --visualize*
  measure **CVs** using ***tuneCV*** (***carputils***) and visualize using ***meshalyzer***

- *tissue_calibrate.py --plan plan.json --resolution 400*
  --restitute-pcl 600 550 500 450 … 320 300
  --plot-cv-restitution
  plot **CV** restitution curves as a function of **PCL**

- *tissue_calibrate.py --plan plan.json --resolution 400*
  --dx-list 50 100 200 300 … 600 700 --plot-cv-dx
  plot **CV** restitution curves as a function of the mesh resolution

- *tissue_calibrate.py --plan plan.json --converge*
  --resolution 400 …
  adjust conductivities to achieve reference velocities
  - *--tune-mode gi*
    adjust intracellular conductivity only
  - *--tune-mode gm*
    adjust intracellular and extracellular conductivities
  - *--update*
    update the conductivity sections in the plan file

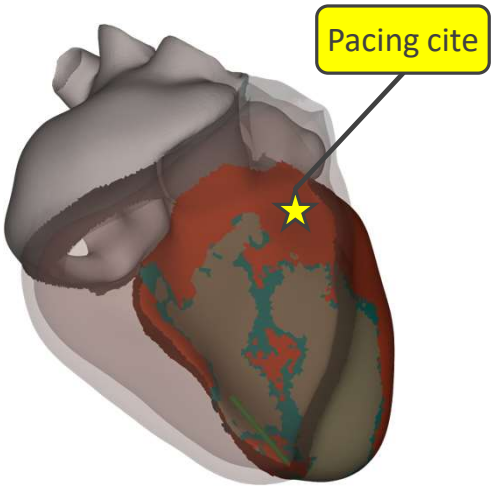
Stage 3: Tissue Limit Cycle

In this stage we aim to find a stable limit cycle at tissue level, i.e.

$$\boldsymbol{\eta}_p(\mathbf{x}, t) = \boldsymbol{\eta}_p(\mathbf{x}, t + T_{cl})$$

for $t \geq T_{stab} > T_{cl}$ and $\mathbf{x} \in \Omega \subset \mathbb{R}^d, d = 2,3.$

Several approaches to approximate the tissue limit cycle are implemented in the ***ForCEPSS*** framework:



- *No pre-pacing*:
  The tissue is not initialized an **EP** state is governed by the model's default initial state.
- *Cellular limit cycle*:
  The initial state vector encoding the cellular limit cycle is used to initialize the experiment.
- ***LAT** pre-pacing:*
  Cell state $\boldsymbol{\eta}_p$ for each location $\mathbf{x} \in \boldsymbol{\Omega}$ is determined by the local activation time as
  $$\boldsymbol{\eta}_p = \boldsymbol{\eta}_p\big(\mathbf{x}, -t_a(\mathbf{x})\big)$$
  optionally followed by several full reaction diffusion cycles.
- *Full physics pre-pacing*:
  A state vector, either the default one or a cellular limit cycle, is used to run several cycles in full physics reaction diffusion mode.

Stage 3: Tissue Limit Cycle

```
"configurations": {
     "version": 2,
     "definitions": {
          "healthy": {
               "tags": [1],
               "func": "ventricle-norm“
          },
          "bz": {
               "tags": [2],
               "func": "ventricle-bz“
          },
          "isthmus": {
               "tags": [3],
               "func": "ventricle-bz“
          },
          "scar": {
               "tags": [4],
               "func": "scar“
          }
     }
}
```

- *configurations section*
  Link tag-regions to **EP** functions

Stage 3: Tissue Limit Cycle

```
"electrodes": {
     "version": 2,
     "definitions": {
          "elec0": {
               "type": "vtxlist",
               "vtxfile": "path/to/vtxfile"
          },
          "elec1": {
               "type": "vtxdata",
               "vtxdata": [1,4,…,123]
          },
          "elec2": {
               "type": "tag",
               "tag": 7
          },
          "elec3": {
               "type": "sphere",
               "p0": [1000, -500, 100],
               "radius": 1000
          },
          ...
     }
}
```

- *configurations section*
  Link tag-regions to **EP** functions

- *electrodes section*
  Define electrodes in an abstract way
  - vtxlist (vertex file)
  - vtxdata (vertex list)
  - tag
  - sphere
  - cylinder
  - block

Stage 3: Tissue Limit Cycle

```
"protocols": {
     "version": 2,
     "apd_restitution": {
          ...
     },
     "prepacing": {
          "protocol0": {
               "propagation": "rd",
               "num_cycles": 2,
               "bcl": 600.0,
               "electrodes": ["elec0"],
               "rel_timings": [0.0],
               "lat_file": null,
               "restart": null,
          },
          "protocol1": {
               "propagation": "ek",
               "num_cycles": 2,
               "bcl": 600.0,
               "electrodes": ["elec1"],
               "rel_timings": [10.0],
               "lat_file": null,
               "restart": null,
          },
          ...
     }
}
```

- *configurations section*
  Link tag-regions to **EP** functions

- *electrodes section*
  Define electrodes in an abstract way
  - vtxlist (vertex file)
  - vtxdata (vertex list)
  - tag
  - sphere
  - cylinder
  - block

- *protocols section*
  - Define **APD**-restitution protocols
  - Define pre-pacing protocols

Stage 3: Tissue Limit Cycle

```
"functions": {
      ...
},
"configurations": {
      ...
},
"electrodes": {
      ...
},
"protocols": {
      ...
},
"solver_setup": {
      ...
}
```

- *limit_cycle.py --plan plan.json --protocols prtcl.json*
  --electrodes elecs.json --protocol protocol0
  --mesh isthgeom --stage gen-lat --gen-lat rd
  --update --visualize

  generate local activation times (**LATs**) (reaction-diffusion based)

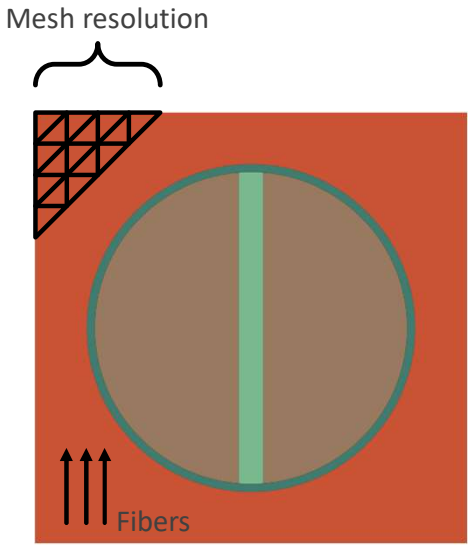
- *limit_cycle.py --plan plan.json --protocols prtcl.json*
  --electrodes elecs.json --protocol protocol0
  --mesh isthgeom --stage lim-cyc --lim-cyc lat-2
  --update --visualize

  compute limits cycle using the computed activation sequence
  - *off*:
    use single cell initial state
  - *lat*:
    use **LAT**-defined activation sequence
  - *lat-n*:
    use **LAT**-defined activation sequence + n full activation sequences
  - *full*:
    full pre-pacing at tissue scale

Example: VARP Protocol

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

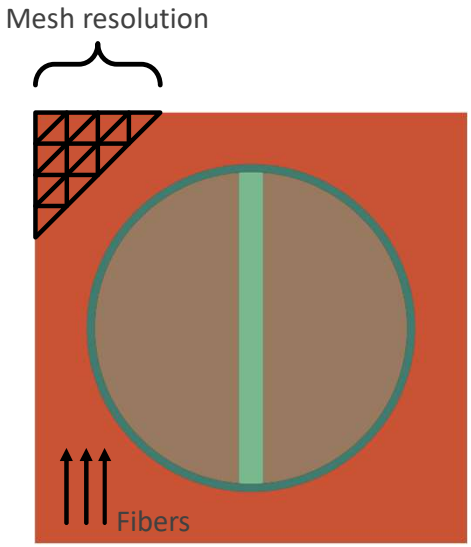
- Given geometry
  - Fiber direction
  - Mesh resolution

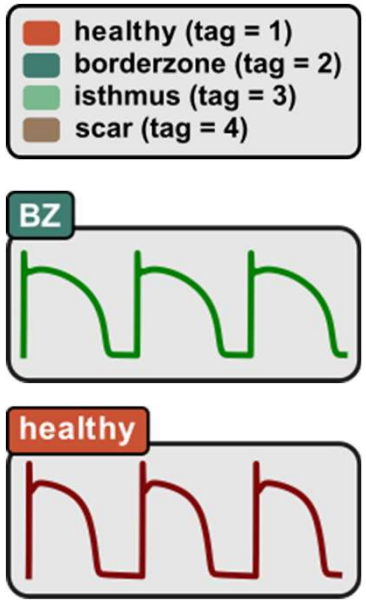


Example: VARP Protocol

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- Given geometry
  - Fiber direction
  - Mesh resolution
- Define tissue regions
- Assign ionic model to regions

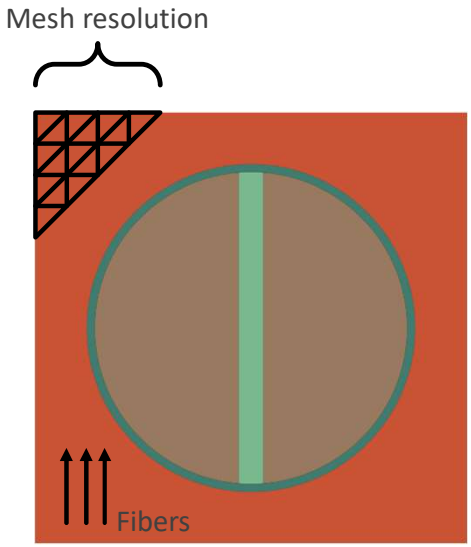


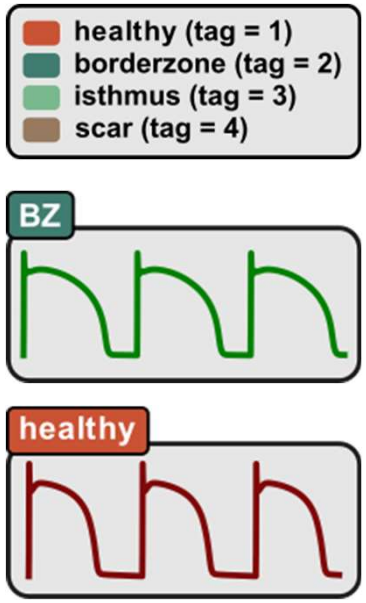


Example: VARP Protocol

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- Given geometry
  - Fiber direction
  - Mesh resolution
- Define tissue regions
- Assign ionic model to regions
- Initialize ionic models
- Tissue calibration

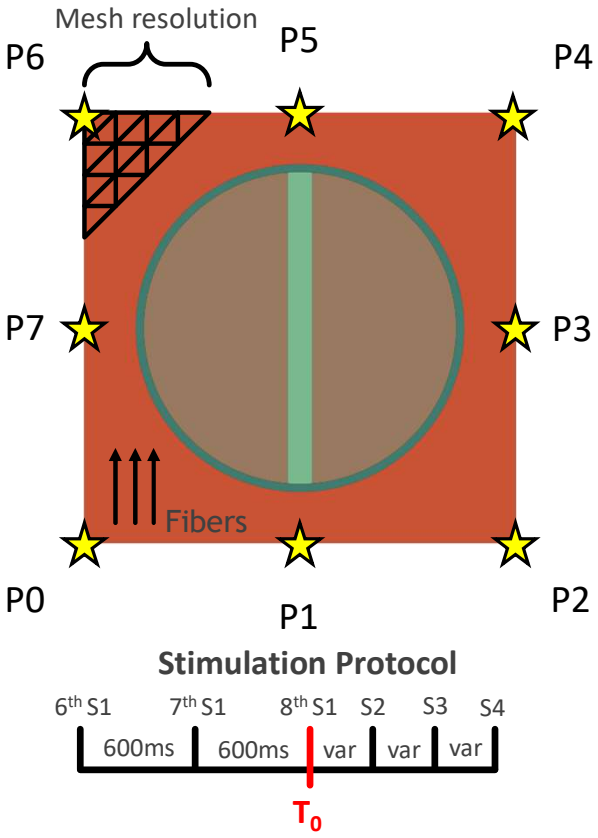


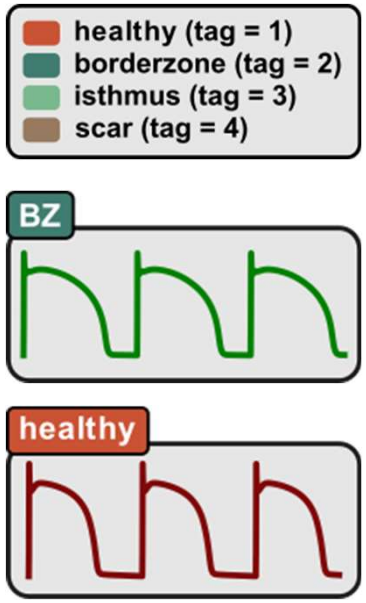


Example: VARP Protocol

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- Given geometry
  - Fiber direction
  - Mesh resolution
- Define tissue regions
- Assign ionic model to regions
- Initialize ionic models
- Tissue calibration
- Define pacing experiment
  - Electrodes
  - Protocols





Task 1: EP Initialization

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- The ***ep_init.py*** script is based on ***bench.opt***, ***carputils***, and ***limpetgui***.
  - To list all available ionic models run
    "bench.opt --list-imp"
  - To list all available ionic model parameters run
    "bench.opt --imp=tenTusscherPanfilov --imp-info"

- **Task1.1:** Setup JSON based plan file defining the following **EP** functions
  - *"ventricle-norm"*
  - *"ventricle-bz"*
  - *"scar"*

  and add an **APD** restitution protocol named *"s1s2"*.

- **Task1.2:** Run ***ep_init.py*** and …
  - visualize 100 cycles of each ionic model with a cycle length of 600ms.
  - check whether you have reached a limit cycle or not.
  - update the simulation plan file in case you have reached a limit cycle.
  - plot **APD** restitution curve.

Task 1: EP Initialization

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

EP-functions:

- "*ventricle-norm*":
  - Ionic model: "tenTusscherPanfilov"
  - Conductivity intra: (0.17, 0.019, 0.019) (gil, git, gin)
  - Conductivity extra: (0.62, 0.24, 0.24) (gel, get, gen)
  - Conduction velocity: (0.853, 0.473, 0.473) (vf, vs, vn)
  - Surface to volume ratio: 0.14
- "*ventricle-bz*":
  - Ionic model: "tenTusscherPanfilov"
  - Model parameters: "GNa-62%,GCaL-69%,GKr-70%,GKs-80%"
  - Conductivity intra: (0.17, 0.019, 0.019) (gil, git, gin)
  - Conductivity extra: (0.62, 0.24, 0.24) (gel, get, gen)
  - Conduction velocity: (0.853, 0.473, 0.473) (vf, vs, vn)
  - Surface to volume ratio: 0.14
- "*scar*":
  - Conductivity: 0.2

Task 1: EP Initialization

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

APD-restitution:

► "*s1s2*":

- basic cycle length of 600ms.
- 5 pre-pacing beats before the pacing protocol.
- 20 beats preceding the premature one.
- start and end coupling interval of 600ms and 50ms, respectively.
- Decrement in S2 prematurity of 10ms.

Task 2: Tissue Calibration

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- The tissue calibration script ***tissue_calibrate.py*** is based on the ***tuneCV*** script which is part of the ***carputils*** framework and utilizes the ***openCARP.opt*** simulator and ***meshalyzer*** for visual inspection.

For the 1D case with an axis ζ the bidomain equation reads

$$\frac{\sigma_{m\zeta}}{\beta}\,\frac{\partial^2 V_m}{\partial \zeta^2} = C_m\,\frac{\partial V_m}{\partial t} + I_{\mathrm{ion}}(V_m, \boldsymbol{\eta})$$

and for a uniformly travelling wave front, space ζ and time t are related by $\zeta = v_\zeta \cdot t$

and we get

$$\frac{\sigma_{m\zeta}}{\beta\, {v_\zeta}^2}\,\frac{\partial^2 V_m}{\partial t^2} = C_m\,\frac{\partial V_m}{\partial t} + I_{\mathrm{ion}}(V_m, \boldsymbol{\eta})\,.$$

If C_m remains unchanged, then V_m remains to be a solution as long $\frac{\sigma_{m\zeta}}{\beta\, {v_\zeta}^2}$ is constant.

$$v_\zeta \propto \sqrt{\frac{\sigma_{m\zeta}}{\beta}}$$

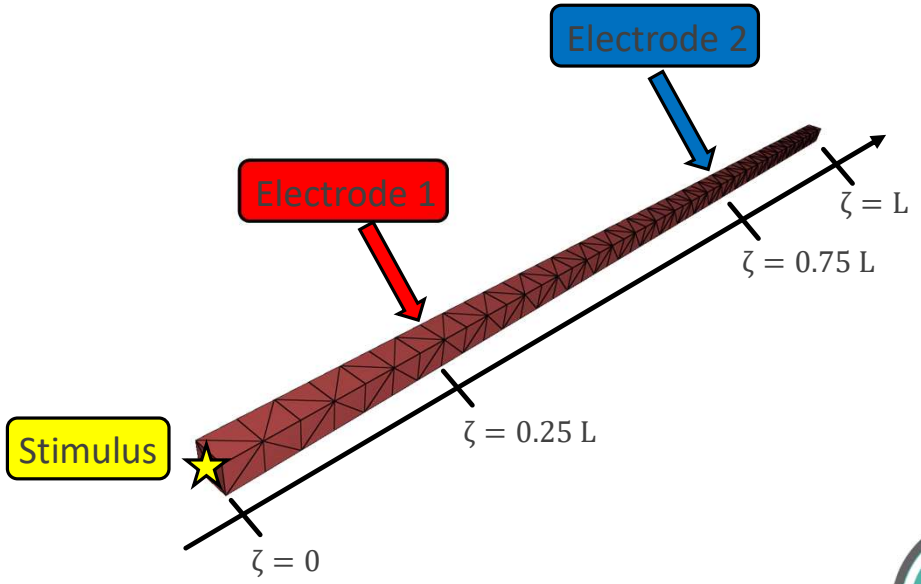
Task 2: Tissue Calibration

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- The tissue calibration script ***tissue_calibrate.py*** is based on the ***tuneCV*** script which is part of the ***carputils*** framework and utilizes the ***openCARP.opt*** simulator and ***meshalyzer*** for visual inspection.

- Create a strand mesh of prescribed resolution with a total length of $\mathrm{L} = 2\ \mathrm{cm}$.

- Apply stimulus at $\zeta = 0$.



Task 2: Tissue Calibration

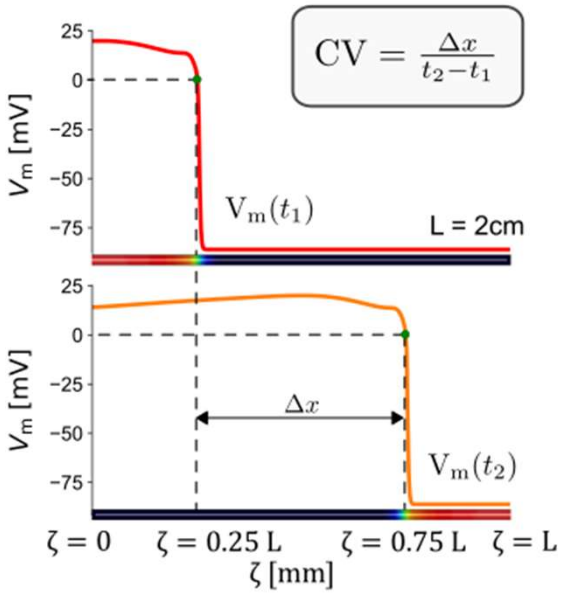
Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- The tissue calibration script ***tissue_calibrate.py*** is based on the ***tuneCV*** script which is part of the ***carputils*** framework and utilizes the ***openCARP.opt*** simulator and ***meshalyzer*** for visual inspection.

- Create a strand mesh of prescribed resolution with a total length of $\mathrm{L} = 2\ \mathrm{cm}$.

- Apply stimulus at $\zeta = 0$.

- Measure arrival time t_1 at $\zeta = 0.25\ \mathrm{L}$ and t_2 at $\zeta = 0.75\ \mathrm{L}$ and compute conduction velocity as $v_\zeta = \frac{0.5\ \mathrm{L}}{t_2 - t_1}$



Task 2: Tissue Calibration

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- The tissue calibration script ***tissue_calibrate.py*** is based on the ***tuneCV*** script which is part of the ***carputils*** framework and utilizes the ***openCARP.opt*** simulator and ***meshalyzer*** for visual inspection.

- **Task2:** Use the updated JSON file from Task1 and run ***tissue_calibrate.py*** to …
  - measure the conduction velocity with the prescribed conductivities and a mesh resolution of 400µm.
  - plot **CV** restitution curves as a function of **PCL**, use PCL = [600, 550, 500, 450, 400, 350, 300].
  - plot **CV** restitution curves as a function of mesh resolution, use dx = [50, 100, 200, 300, 400, 500].
  - tune the intracellular conductivities only to match the reference **CVs**.
  - tune the intracellular & extracellular conductivities to match the reference **CVs**. and update the simulation plan.

Task 3: Tissue Limit Cycle

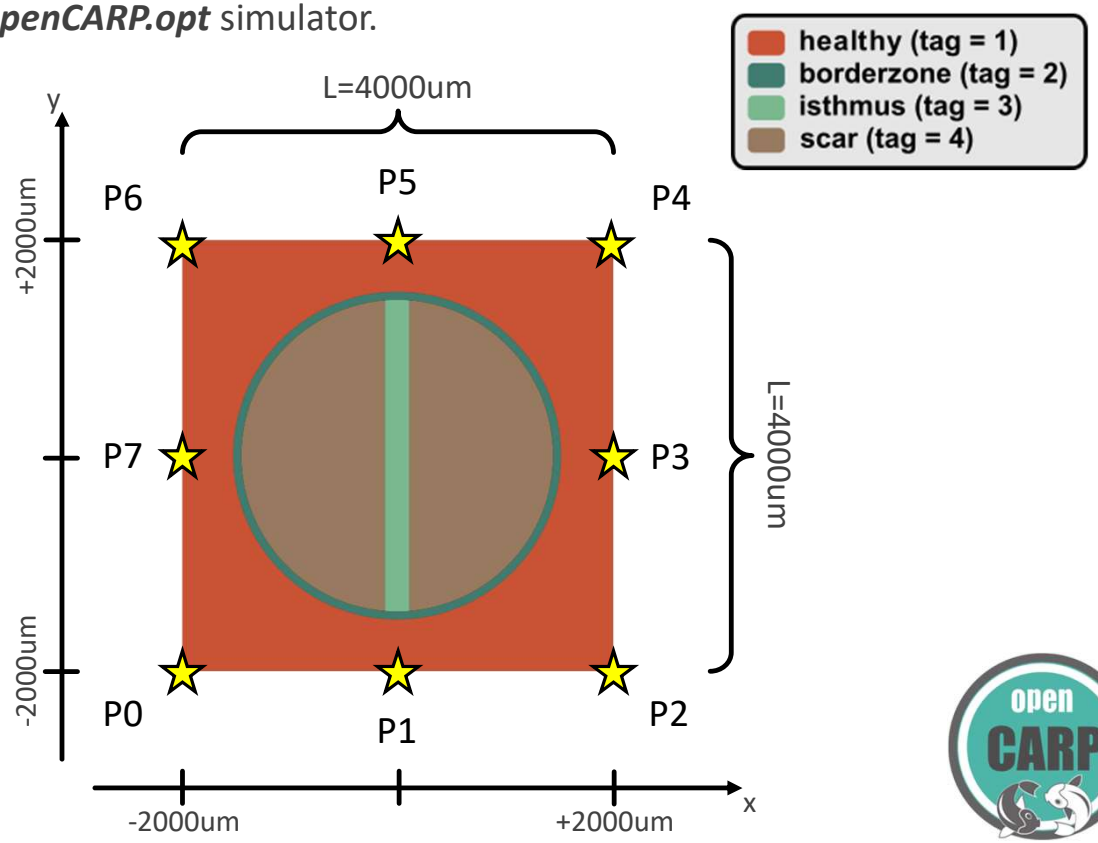
Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- The tissue limit cycle computation utilizes the ***openCARP.opt*** simulator.

- **Task3.1:** Extend the plan file and add the configurations
  - *"ventricle-norm"*
  - *"ventricle-bz"*
  - *"scar"*

  linked to the corresponding EP function.

- **Task3.2:** Create the file *elec.json* defining the 8 pacing electrodes as spheres with a radius of 1000um.



Task 3: Tissue Limit Cycle

Implement the VARP (*Virtual Arrhythmia Risk Prediction*) protocol using ***ForCEPSS***

- The tissue limit cycle computation utilizes the ***openCARP.opt*** simulator.
- **Task3.3:** Create the file *prtcl.json* and add a pre-pacing protocol named "*lcP1*" with the following setup:
  - use reaction-diffusion propagation model,
  - pre-pacing cycle length of 600ms
  - three pre-pacing cycles
  - the electrode "*P1*" with a start time at 0ms
- **Task3.4**: Use the updated JSON file from Task1 and *elecs.json as well as prtcls.json* to run the ***limit_cycle.py*** script to …
  - create and visualize local activation times using a reaction-diffusion model and the protocol "*lcP1*", update the plan file
  - create tissue limit cycle of type "*lat-1*" using protocol "*lcP1*" and visualize the result

Final remarks

Thank you for your attention and interest in ***ForCEPSS*** and ***openCARP*** in general.

- Please use the software and tools we have developed.
- The more users, the more feedback we receive.
- The more feedback, the better and more robust we can make the software.

Questions?

matthias.gsell@medunigraz.at