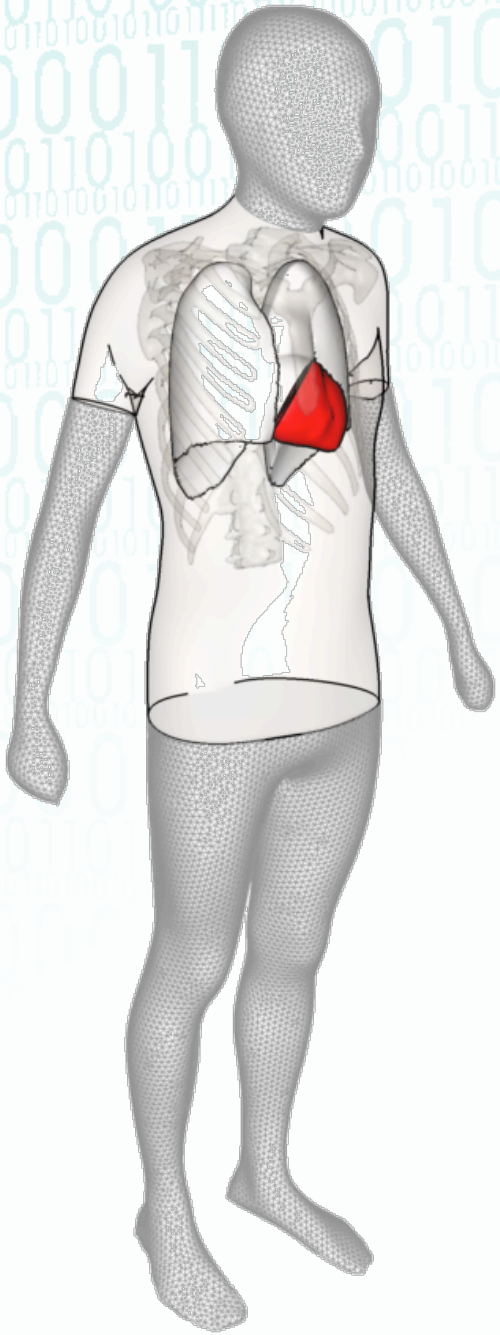
HPC simulations of cardiac electrophysiology using patient-specific models of the human heart

CompBioMed & VPH Institute webinar



Francesc Levrero-Florencio & Ana Mincholé

Computational Cardiovascular Science Group

Department of Computer Science, University of Oxford

CompBioMed

European Commission

VPH Institute

Building the Virtual Physiological Human

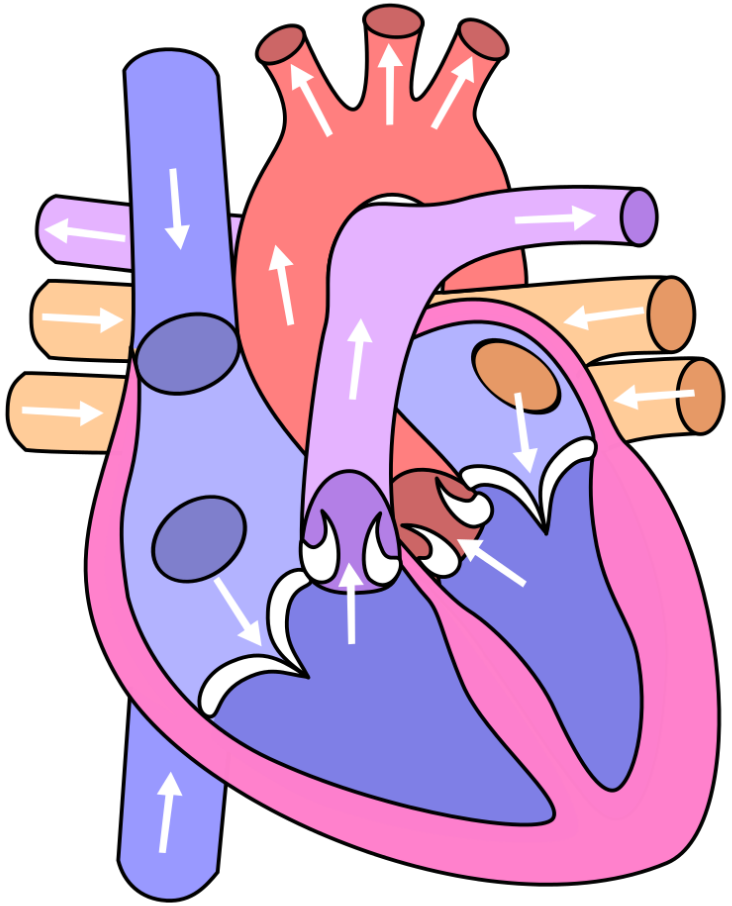
Contents

- Introduction to the heart (electro)physiology
- Mathematical modelling
- CHASTE
- Alya
- Final Comments

Brief introduction to (electro)physiology

Introduction to the physiology of the heart

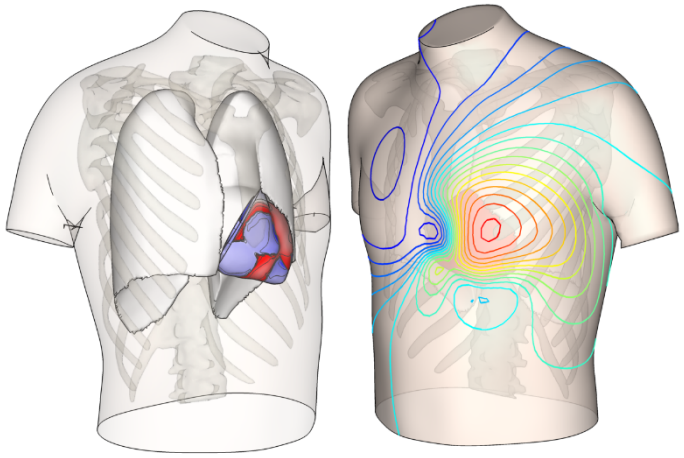
- Heart is a contracting muscle
- 4 chambers
- 2 systems: pulmonary and systemic

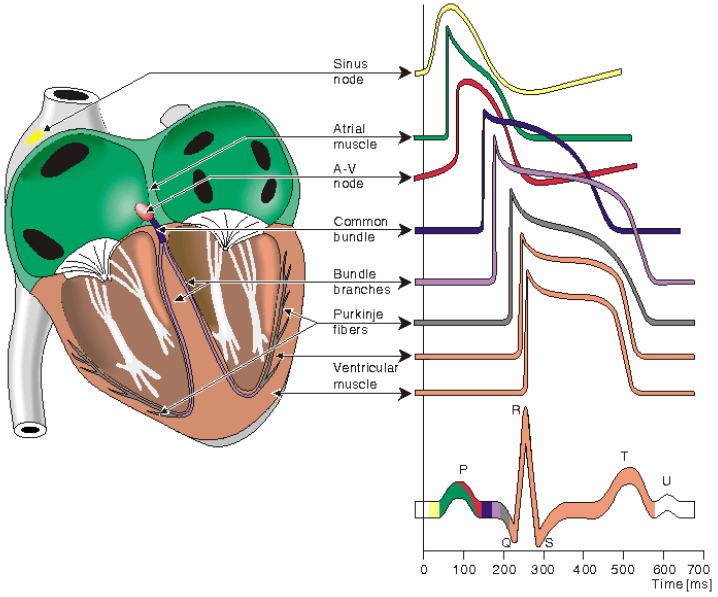


{https://commons.wikimedia.org}

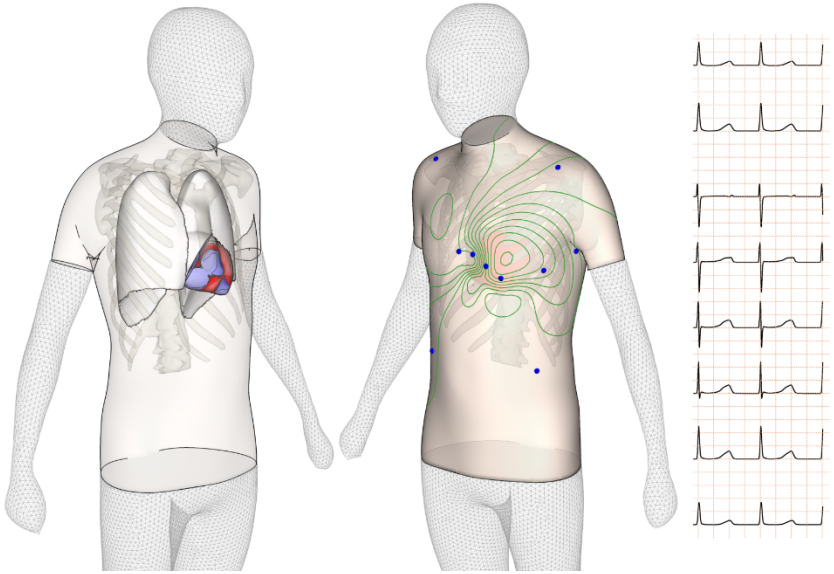
Electrophysiology of the heart

- SA node – Atria - AV node - brief delay - His bundle branches down the septum
- Purkinje fibers allow propagation throughout the endocardium
- Cell to cell propagation through gap junctions.





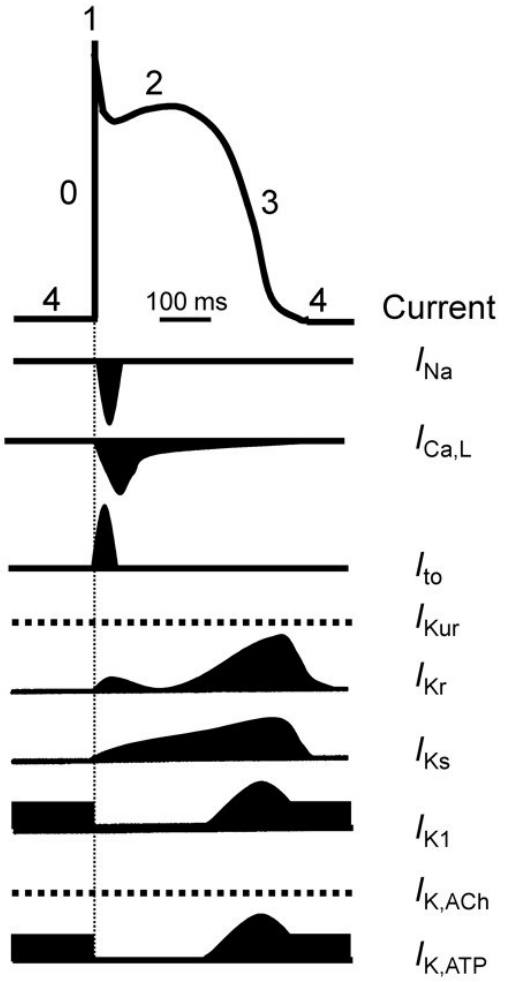
{http://www.bem.fi/book/06/06.htm}



Zacur et al, BIVPCS 2017

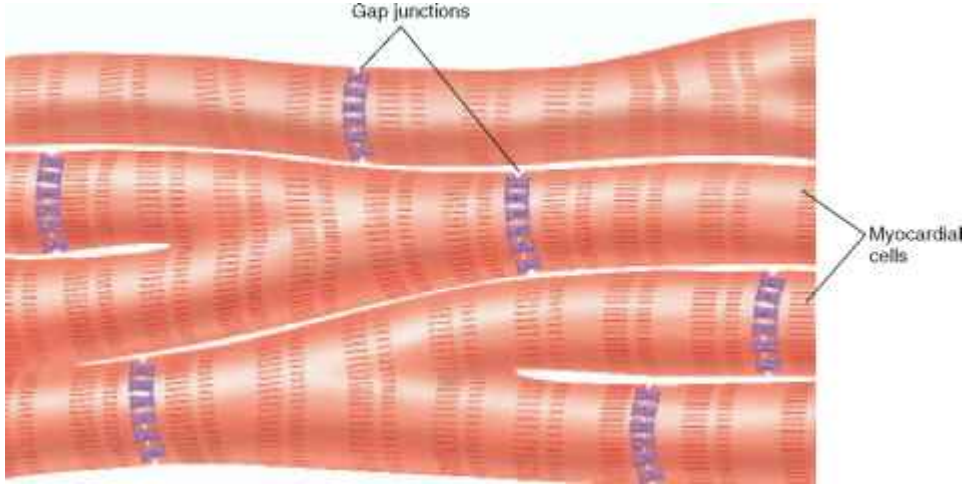
Cell electrophysiology

- Four phases of the action potential:
  - ✓ Upstroke (0/1)
  - ✓ Plateau (2)
  - ✓ Repolarisation (3)
  - ✓ Resting potential (4)
- Calcium and Sodium currents are involved during the upstroke and plateau phases
- Potassium-based currents are involved in the repolarisation

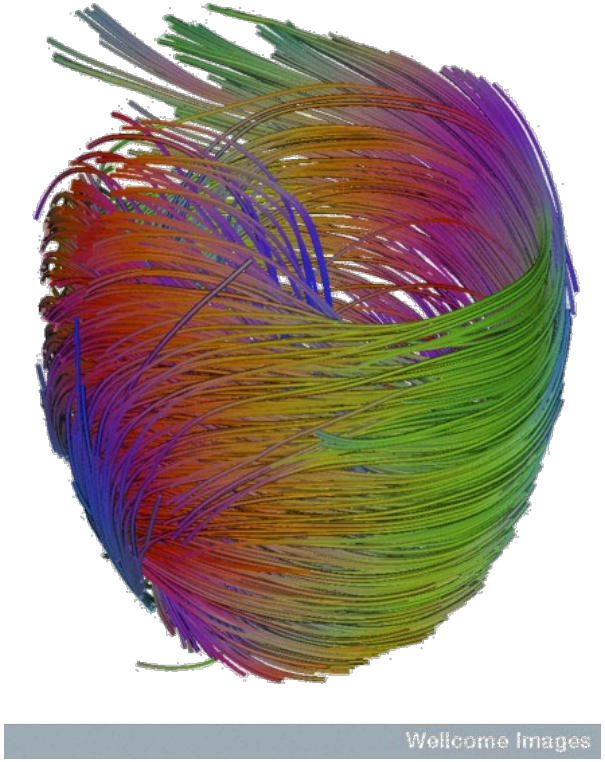


{Ravens U and Cerbai E. Europace 2008}

Tissue electrophysiology



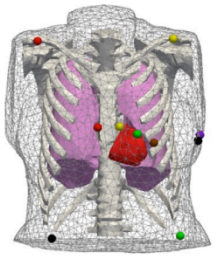
{Essentials of Human Physiology 2010}

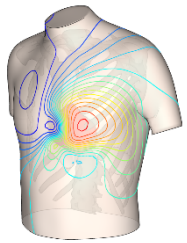


Cardiac modelling

MULTISCALE INTEGRATION

Body
- Body surface potentials
- ECG recordings
- mHealth data



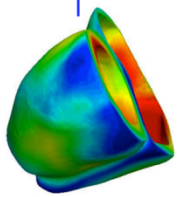


Propagation model

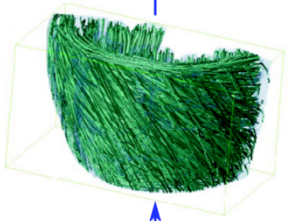
$$\nabla \cdot \overline{\sigma}_i \nabla \Phi_i = \beta \cdot I_m - I_{st\_i}$$
$$\nabla \cdot \overline{\sigma}_e \nabla \Phi_e = -\beta \cdot I_m$$

archer

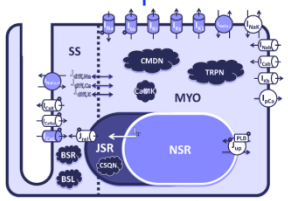
Organ
- Clinical EP studies
- Multi-modality clinical MRI



Tissue
- High-resolution MRI
- Histology
- Optical mapping
- Cell cultures

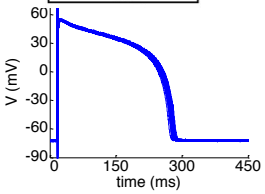


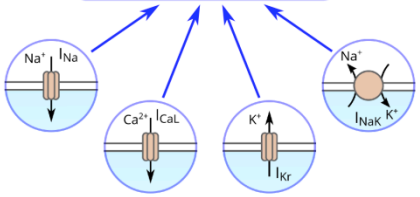
Single cell
- Microelectrode recordings
- Protein/mRNA expression
- Optical mapping



AP cell model

$$I_m = C\frac{dV_m}{dt} + I_{ion}$$





Ionic current
- Voltage/patch clamp
- Isolated cells/hiPSC-CMs

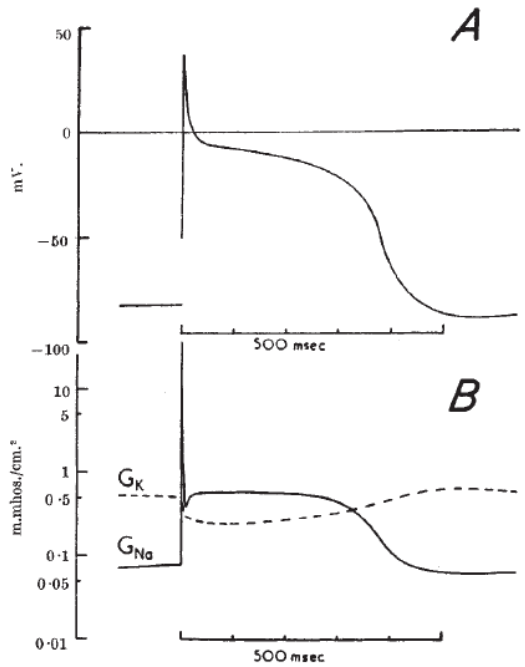
Ionic currents

$$I_{Na} = g_{Namax} * m^3 * h * j * (V_m - E_{Na})$$
$$\frac{dx}{dt} = \alpha_x(1-x) - \beta_x x$$
$$\alpha_x = \alpha_x(V_m); \beta_x = \beta_x(V_m)$$

Mathematical modelling

First cardiac AP model

Denis Noble
(1936-present)



No 4749 November 5, 1960 NATURE 495

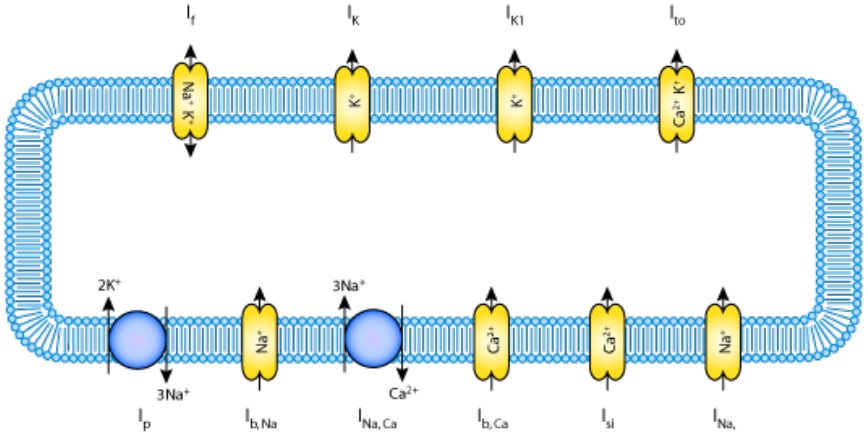
Cardiac Action and Pacemaker Potentials based on the Hodgkin–Huxley Equations

D. Noble

Phil. Trans. R. Soc. Lond. B **307**, 353–398 (1985) [353]
Printed in Great Britain

A MODEL OF CARDIAC ELECTRICAL ACTIVITY INCORPORATING IONIC PUMPS AND CONCENTRATION CHANGES

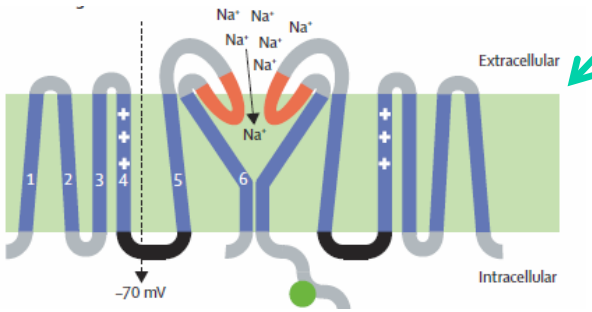
By D. DiFrancesco[1] and D. Noble, F.R.S.[2]



… to current human AP models

- Based on experimental data from **>150 human hearts**.
- Still largely based on Hodgkin-Huxley model and its formulation of voltage-gated ion channel behaviour.

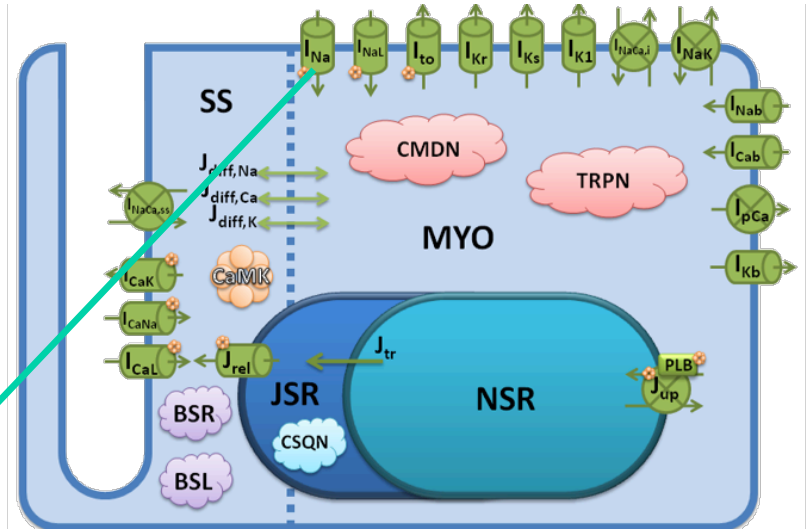
$$C_m \frac{dV_m}{dt} = I_{stim} - I_{currents}$$



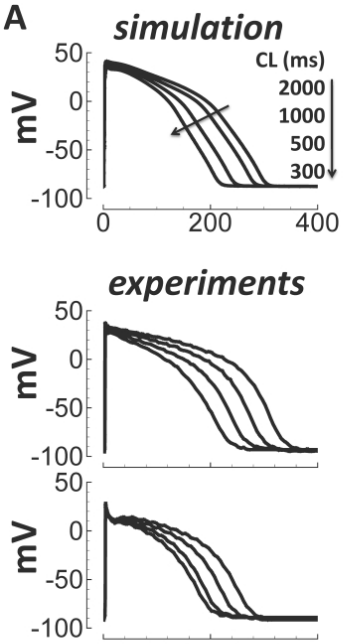
Sodium channel schematic

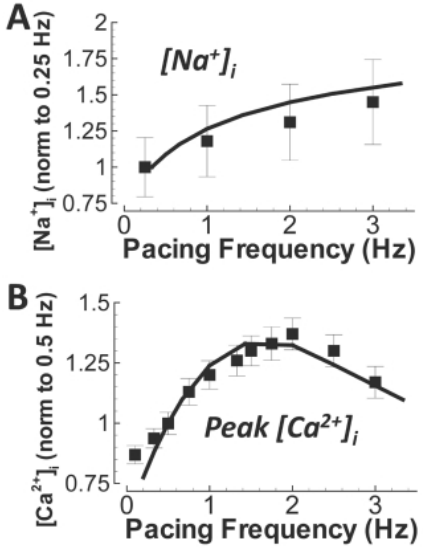
Sodium channel equation

$$I_{Na} = g_{Na} m^3 h (V_m - E_{Na})$$



{O'Hara et al. PLOS Comput Biol. 2011}





Monodomain and bidomain models

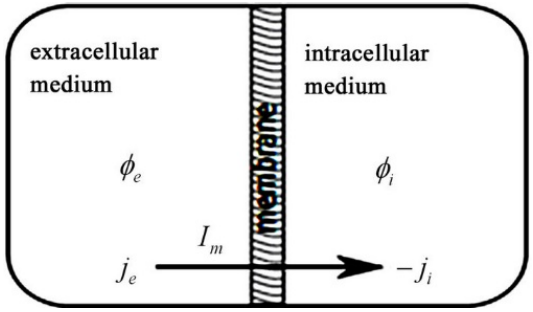
Monodomain model

$$\nabla \cdot (\mathbf{D}_0 \nabla V) = \chi (C \frac{\mathrm{d}V}{\mathrm{d}t} + I_{ion} - I_{app})$$

Bidomain model

$$\nabla \cdot (\mathbf{D}_{int} \nabla V) + \nabla \cdot (\mathbf{D}_{int} \nabla U) = \chi (C \frac{\mathrm{d}V}{\mathrm{d}t} + I_{ion} - I_{app})$$

$$\nabla \cdot [(\mathbf{D}_{int} + \mathbf{D}_{ext}) \nabla U] + \nabla \cdot (\mathbf{D}_{int} \nabla V) = 0$$



{Adebisi et al. J Biomed Sci Eng. 2013}

Conductivity Tensors

- The ratio of the intracellular and extracellular conductivity tensors;

Bidomain: $\frac{D_{\parallel}^i}{D_{\parallel}^e} \neq \frac{D_{\perp}^i}{D_{\perp}^e}$

Monodomain: $\frac{D_{\parallel}^i}{D_{\parallel}^e} = \frac{D_{\perp}^i}{D_{\perp}^e} = \alpha$

$$\mathbf{D}_i = \begin{pmatrix} D_{\parallel}^i & 0 & 0 \\ 0 & D_{\perp}^i & 0 \\ 0 & 0 & D_{\perp}^i \end{pmatrix}$$

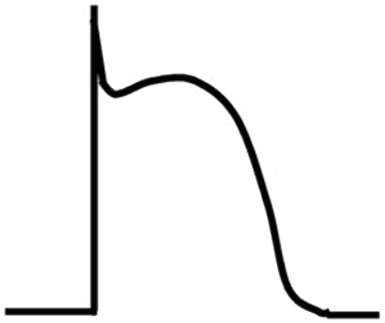
$$\mathbf{D}_e = \begin{pmatrix} D_{\parallel}^e & 0 & 0 \\ 0 & D_{\perp}^e & 0 \\ 0 & 0 & D_{\perp}^e \end{pmatrix}$$

Cardiac tissue is more accurately described as a three-dimensional anisotropic *bidomain*, especially under conditions of applied external current such as in defibrillation studies.[1-2]

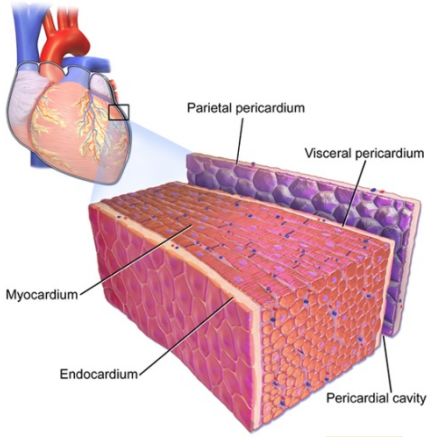
[1] B. J. Roth and J. P. Wikswo, IEEE Transactions on Biomedical Engineering **41**, 232-240 (1994)
[2] J. P. Wikswo, *et al.*, Biophysical Journal **69**, 2195-2210 (1995)

Integrative physiology through modelling

Cellular Electrophysiology

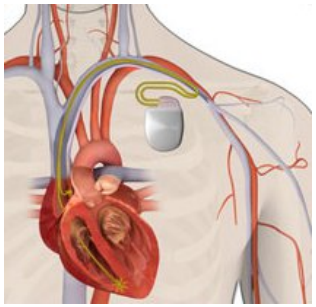


Tissue Properties



Membrane Kinetics

Electrical Stimulation



$$C_m \frac{\partial V_m}{\partial t} = \frac{1}{S_v} \nabla \cdot \sigma \nabla V_m - I_{ion} + I_{stim}$$

Propagation of the electrical impulse

Adapted from Dr Vincent Jacquemet (Univ. Montreal, Canada)

Considered simulation software

- ✓ 0D,1D, 2D and 3D
- ✓ Petsc and Metis
- ✓ C++
- ✓ FE, BE, RK, CVODE
- ✓ Multi-scale simulation
- ✓ Highly scalable

- ✓ 0D,1D, 2D and 3D
- ✓ In-house and Metis
- ✓ Modern Fortran
- ✓ FE
- ✓ Multiphysics
- ✓ Highly scalable (up to 100k cores)

DEPARTMENT OF
COMPUTER
SCIENCE

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Example of electrophysiology with Chaste

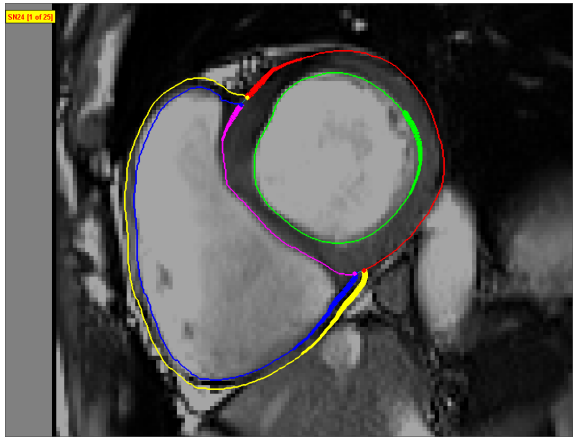
Chaste

- Cardiac Chaste functionalities:
  - ✓ Monodomain and bidomain models
  - ✓ Automatic implementation of cellular action potential models from the CellML repository
  - ✓ Automatic generation of mathematical model for fibre orientation
  - ✓ Checkpoint of simulations midway through run and restart with altered parameters
  - ✓ Post-processing of simulation results to calculate electrophysiological properties such as action potential duration, conduction velocity, etc.

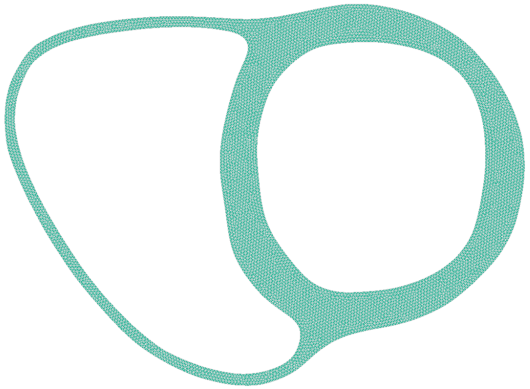
http://www.cs.ox.ac.uk/chaste/cardiac_index.html

2D electrical propagation using Chaste

- 2D Mesh geometry

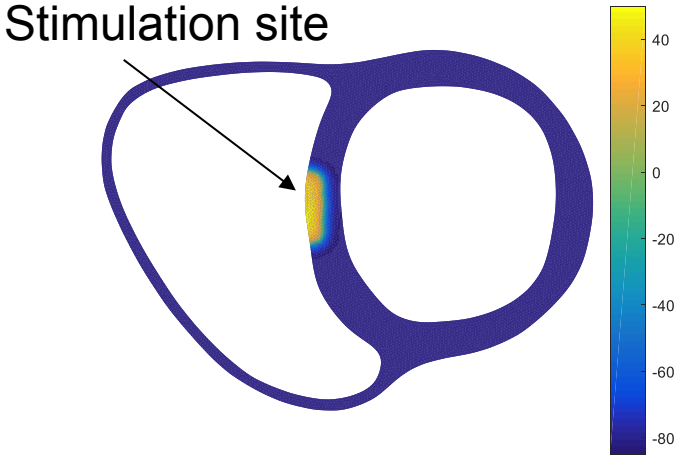


Courtesy of Dr. Rina Ariga



Courtesy of Dr. Ernesto Zacur

- Control 2D Electrical propagation



- 2D tissue from MRI
- O'Hara Rudy 2011 epicardial model
- Bidomain model
- Isotropic conductivities
- Regular stimulus of 600 ms

```cpp
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"

class MyCellFactory : public AbstractCardiacCellFactory<2>
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus;

public:
    MyCellFactory(float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0, 1, period, 0))
    {
    }

    AbstractCvodeCell* CreateCardiacCellForTissueNode(Node<2>* pNode)
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_cell;

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) && (y<0.1) )
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpStimulus);

        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
        }
        return p_cell;
    }
};

class TestConductivity : public CxxTest::TestSuite
{
public:
    void TestConductivity2d() throw(Exception)
    {
        // MESH
        std::string filepath = "projects/anamin/test/data/2Dmesh";
        DistributedTetrahedralMesh<2,2> mesh;
        TrianglesMeshReader<2,2> mesh_reader(filepath);
        mesh.ConstructFromMeshReader(mesh_reader);

        // NUMERICS
        double odeT, pdeT, Print_Time;
        odeT =  0.025;
        pdeT =  0.05;
        Print_Time=1;  // ms

        HeartConfig::Instance()->SetOdePdeAndPrintingTimeSteps(odeT, pdeT, Print_Time);

        // BIDOMAIN PARAMETERS
        HeartConfig::Instance()->SetSurfaceAreaToVolumeRatio(1400);     // 1/cm
        HeartConfig::Instance()->SetCapacitance(1.0);                   // uF/cm^2
        HeartConfig::Instance()->SetIntracellularConductivities(Create_c_vector(1.2, 1.2));
        HeartConfig::Instance()->SetExtracellularConductivities(Create_c_vector(4, 4));

        // SIMULATION TIME
        HeartConfig::Instance()->SetSimulationDuration(1300); //ms

        //POSTPROCESSING OPTIONS
        // APD map
        std::vector<std::pair<double,double> > apd_map;
        apd_map.push_back(std::pair<double, double>(90.0, 0.0));    // APD90, 0 mV threshold
        HeartConfig::Instance()->SetApdMaps(apd_map);

        // Activation time map
        std::vector<double> upstroke_time_map;
        upstroke_time_map.push_back(0.0);                           // 0 mV threshold
        HeartConfig::Instance()->SetUpstrokeTimeMaps(upstroke_time_map);

        // Max Upstroke Velocity map
        std::vector<double> maxupstroke_vel_map;
        maxupstroke_vel_map.push_back(0.0);
        HeartConfig::Instance()->SetMaxUpstrokeVelocityMaps(maxupstroke_vel_map);


        // OUTPUT OPTIONS
        HeartConfig::Instance()->SetOutputDirectory("./OUT");
        HeartConfig::Instance()->SetOutputFilenamePrefix("results");
        HeartConfig::Instance()->SetOutputUsingOriginalNodeOrdering(true);
        HeartConfig::Instance()->SetVisualizeWithMeshalyzer(false);
        HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

        // PARAMETERS
        double period           = 600.0;

        MyCellFactory cell_factory(period);

        // PROBLEM
        BidomainProblem<2> bidomain_problem( &cell_factory );

        bidomain_problem.SetMesh(&mesh);
        bidomain_problem.SetWriteInfo();
        bidomain_problem.Initialise();
        bidomain_problem.Solve();
    }
};
```

```
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"
```

```
class MyCellFactory : public AbstractCardiacCellFactory<2>
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus;

public:
    MyCellFactory(float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0, 1, period, 0))
    {
    }

    AbstractCvodeCell* CreateCardiacCellForTissueNode(Node<2>* pNode)
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_cell;

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) && (y<0.1) )
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpStimulus);

        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
        }
        return p_cell;
    }
};
```

```
class TestConductivity : public CxxTest::TestSuite
{
public:
    void TestConductivity2d() throw(Exception)
    {
        // MESH
        std::string filepath = "projects/anamin/test/data/2Dmesh";
        DistributedTetrahedralMesh<2,2> mesh;
        TrianglesMeshReader<2,2> mesh_reader(filepath);
        mesh.ConstructFromMeshReader(mesh_reader);

        // NUMERICS
        double odeT, pdeT, Print_Time;
        odeT =  0.025;
        pdeT =  0.05;
        Print_Time=1;  // ms

        HeartConfig::Instance()->SetOdePdeAndPrintingTimeSteps(odeT, pdeT, Print_Time);

        // BIDOMAIN PARAMETERS
        HeartConfig::Instance()->SetSurfaceAreaToVolumeRatio(1400);     // 1/cm
        HeartConfig::Instance()->SetCapacitance(1.0);                   // uF/cm^2
        HeartConfig::Instance()->SetIntracellularConductivities(Create_c_vector(1.2, 1.2));
        HeartConfig::Instance()->SetExtracellularConductivities(Create_c_vector(4, 4));

        // SIMULATION TIME
        HeartConfig::Instance()->SetSimulationDuration(1300); //ms

        //POSTPROCESSING OPTIONS
        // APD map
        std::vector<std::pair<double,double> > apd_map;
        apd_map.push_back(std::pair<double, double>(90.0, 0.0));    // APD90, 0 mV threshold
        HeartConfig::Instance()->SetApdMaps(apd_map);

        // Activation time map
        std::vector<double> upstroke_time_map;
        upstroke_time_map.push_back(0.0);                           // 0 mV threshold
        HeartConfig::Instance()->SetUpstrokeTimeMaps(upstroke_time_map);

        // Max Upstroke Velocity map
        std::vector<double> maxupstroke_vel_map;
        maxupstroke_vel_map.push_back(0.0);
        HeartConfig::Instance()->SetMaxUpstrokeVelocityMaps(maxupstroke_vel_map);


        // OUTPUT OPTIONS
        HeartConfig::Instance()->SetOutputDirectory("./OUT");
        HeartConfig::Instance()->SetOutputFilenamePrefix("results");
        HeartConfig::Instance()->SetOutputUsingOriginalNodeOrdering(true);
        HeartConfig::Instance()->SetVisualizeWithMeshalyzer(false);
        HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

        // PARAMETERS
        double period           = 600.0;

        MyCellFactory cell_factory(period);

        // PROBLEM
        BidomainProblem<2> bidomain_problem( &cell_factory );

        bidomain_problem.SetMesh(&mesh);
        bidomain_problem.SetWriteInfo();
        bidomain_problem.Initialise();
        bidomain_problem.Solve();
    }
};
```

```
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"

class MyCellFactory : public AbstractCardiacCellFactory<2>
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus;

public:
    MyCellFactory(float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0, 1, period, 0))
    {
    }

    AbstractCvodeCell* CreateCardiacCellForTissueNode(Node<2>* pNode)
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_cell;

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) && (y<0.1) )
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpStimulus);

        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
        }
        return p_cell;
    }
};

class TestConductivity : public CxxTest::TestSuite
{
public:
    void TestConductivity2d() throw(Exception)
    {
        // MESH
        std::string filepath = "projects/anamin/test/data/2Dmesh";
        DistributedTetrahedralMesh<2,2> mesh;
        TrianglesMeshReader<2,2> mesh_reader(filepath);
        mesh.ConstructFromMeshReader(mesh_reader);

        // NUMERICS
        double odeT, pdeT, Print_Time;
        odeT =  0.025;
        pdeT =  0.05;
        Print_Time=1;  // ms

        HeartConfig::Instance()->SetOdePdeAndPrintingTimeSteps(odeT, pdeT, Print_Time);

        // BIDOMAIN PARAMETERS
        HeartConfig::Instance()->SetSurfaceAreaToVolumeRatio(1400);     // 1/cm
        HeartConfig::Instance()->SetCapacitance(1.0);                   // uF/cm^2
        HeartConfig::Instance()->SetIntracellularConductivities(Create_c_vector(1.2, 1.2));
        HeartConfig::Instance()->SetExtracellularConductivities(Create_c_vector(4, 4));

        // SIMULATION TIME
        HeartConfig::Instance()->SetSimulationDuration(1300); //ms

        //POSTPROCESSING OPTIONS
        // APD map
        std::vector<std::pair<double,double> > apd_map;
        apd_map.push_back(std::pair<double, double>(90.0, 0.0));    // APD90, 0 mV threshold
        HeartConfig::Instance()->SetApdMaps(apd_map);

        // Activation time map
        std::vector<double> upstroke_time_map;
        upstroke_time_map.push_back(0.0);                           // 0 mV threshold
        HeartConfig::Instance()->SetUpstrokeTimeMaps(upstroke_time_map);

        // Max Upstroke Velocity map
        std::vector<double> maxupstroke_vel_map;
        maxupstroke_vel_map.push_back(0.0);
        HeartConfig::Instance()->SetMaxUpstrokeVelocityMaps(maxupstroke_vel_map);


        // OUTPUT OPTIONS
        HeartConfig::Instance()->SetOutputDirectory("./OUT");
        HeartConfig::Instance()->SetOutputFilenamePrefix("results");
        HeartConfig::Instance()->SetOutputUsingOriginalNodeOrdering(true);
        HeartConfig::Instance()->SetVisualizeWithMeshalyzer(false);
        HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

        // PARAMETERS
        double period           = 600.0;

        MyCellFactory cell_factory(period);

        // PROBLEM
        BidomainProblem<2> bidomain_problem( &cell_factory );

        bidomain_problem.SetMesh(&mesh);
        bidomain_problem.SetWriteInfo();
        bidomain_problem.Initialise();
        bidomain_problem.Solve();
    }
};
```


```
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"
```

```
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"

class MyCellFactory : public AbstractCardiacCellF
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus

public:
    MyCellFactory(float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.
    {
    }

    AbstractCvodeCell* CreateCardiacCellForTissue
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) &
        {
            p_cell = new CellORd2011epi_fkatpFrom

        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFrom
        }
        return p_cell;
    }
};

class TestConductivity : public CxxTest::TestSuit
{
public:
    void TestConductivity2d() throw(Exception)
    {
        // MESH
        std::string filepath = "projects/anamin/t
        DistributedTetrahedralMesh<2,2> mesh;
        TrianglesMeshReader<2,2> mesh_reader(file
        mesh.ConstructFromMeshReader(mesh_reader)

        // NUMERICS
        double odeT, pdeT, Print_Time;
        odeT =  0.025;
        pdeT =  0.05;
        Print_Time=1;  // ms

        HeartConfig::Instance()->SetOdePdeAndPrin

        // BIDOMAIN PARAMETERS
        HeartConfig::Instance()->SetSurfaceAreaTo
        HeartConfig::Instance()->SetCapacitance(1
        HeartConfig::Instance()->SetIntracellular
        HeartConfig::Instance()->SetExtracellular

        // SIMULATION TIME
        HeartConfig::Instance()->SetSimulationDur

        //POSTPROCESSING OPTIONS
        // APD map
        std::vector<std::pair<double,double> > ap
        apd_map.push_back(std::pair<double, doubl
        HeartConfig::Instance()->SetApdMaps(apd_m

        // Activation time map
        std::vector<double> upstroke_time_map;
        upstroke_time_map.push_back(0.0);
        HeartConfig::Instance()->SetUpstrokeTimeM

        // Max Upstroke Velocity map
        std::vector<double> maxupstroke_vel_map;
        maxupstroke_vel_map.push_back(0.0);
        HeartConfig::Instance()->SetMaxUpstrokeVe


        // OUTPUT OPTIONS
        HeartConfig::Instance()->SetOutputDirecto
        HeartConfig::Instance()->SetOutputFilenamePrefix("results");
        HeartConfig::Instance()->SetOutputUsingOriginalNodeOrdering(true);
        HeartConfig::Instance()->SetVisualizeWithMeshalyzer(false);
        HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

        // PARAMETERS
        double period          = 600.0;

        MyCellFactory cell_factory(period);

        // PROBLEM
        BidomainProblem<2> bidomain_problem( &cell_factory );

        bidomain_problem.SetMesh(&mesh);
        bidomain_problem.SetWriteInfo();
        bidomain_problem.Initialise();
        bidomain_problem.Solve();
    }
};
```

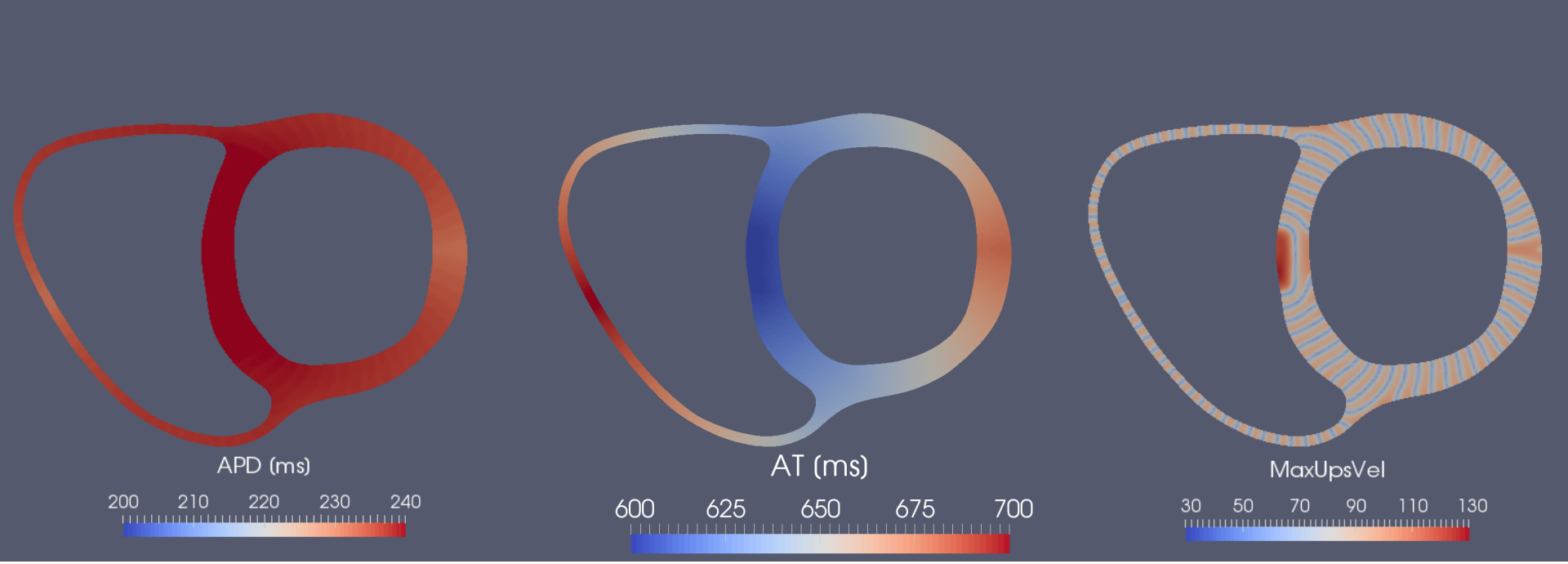
```
class MyCellFactory : public AbstractCardiacCellFactory<2>
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus;

public:
    MyCellFactory(float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0, 1, period, 0))
    {
    }

    AbstractCvodeCell* CreateCardiacCellForTissueNode(Node<2>* pNode)
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_cell;

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) && (y<0.1) )
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpStimulus);

        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
        }
        return p_cell;
    }
};
```

```
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"

class MyCellFactory : public AbstractCardiac
{
private:
    boost::shared_ptr<RegularStimulus> mpStim

public:
    MyCellFactory(float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120
    {
    }

    AbstractCvodeCell* CreateCardiacCellForTi
    {
        CellORd2011epi_fkatpFromCellMLCvodeOp

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.
        {
            p_cell = new CellORd2011epi_fkatp

        }
        else
        {
            p_cell = new CellORd2011epi_fkatp
        }
        return p_cell;
    }
};

class TestConductivity : public CxxTest::Test
{
public:
    void TestConductivity2d() throw(Exceptior
    {
        // MESH
        std::string filepath = "projects/anan
        DistributedTetrahedralMesh<2,2> mesh;
        TrianglesMeshReader<2,2> mesh_reader(
        mesh.ConstructFromMeshReader(mesh_rea

        // NUMERICS
        double odeT, pdeT, Print_Time;
        odeT =  0.025;
        pdeT =  0.05;
        Print_Time=1;  // ms

        HeartConfig::Instance()->SetOdePdeAnd

        // BIDOMAIN PARAMETERS
        HeartConfig::Instance()->SetSurfaceAr
        HeartConfig::Instance()->SetCapacitan
        HeartConfig::Instance()->SetIntracell
        HeartConfig::Instance()->SetExtracell

        // SIMULATION TIME
        HeartConfig::Instance()->SetSimulatio

        //POSTPROCESSING OPTIONS
        // APD map
        std::vector<std::pair<double,double>
        apd_map.push_back(std::pair<double, d
        HeartConfig::Instance()->SetApdMaps(a

        // Activation time map
        std::vector<double> upstroke_time_map
        upstroke_time_map.push_back(0.0);
        HeartConfig::Instance()->SetUpstrokeT

        // Max Upstroke Velocity map
        std::vector<double> maxupstroke_vel_m
        maxupstroke_vel_map.push_back(0.0);
        HeartConfig::Instance()->SetMaxUpstro


        // OUTPUT OPTIONS
        HeartConfig::Instance()->SetOutputDirectory("./OUT");
        HeartConfig::Instance()->SetOutputFilenamePrefix("results");
        HeartConfig::Instance()->SetOutputUsingOriginalNodeOrdering(true);
        HeartConfig::Instance()->SetVisualizeWithMeshalyzer(false);
        HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

        // PARAMETERS
        double period          = 600.0;

        MyCellFactory cell_factory(period);

        // PROBLEM
        BidomainProblem<2> bidomain_problem( &cell_factory );

        bidomain_problem.SetMesh(&mesh);
        bidomain_problem.SetWriteInfo();
        bidomain_problem.Initialise();
        bidomain_problem.Solve();
    }
};
```

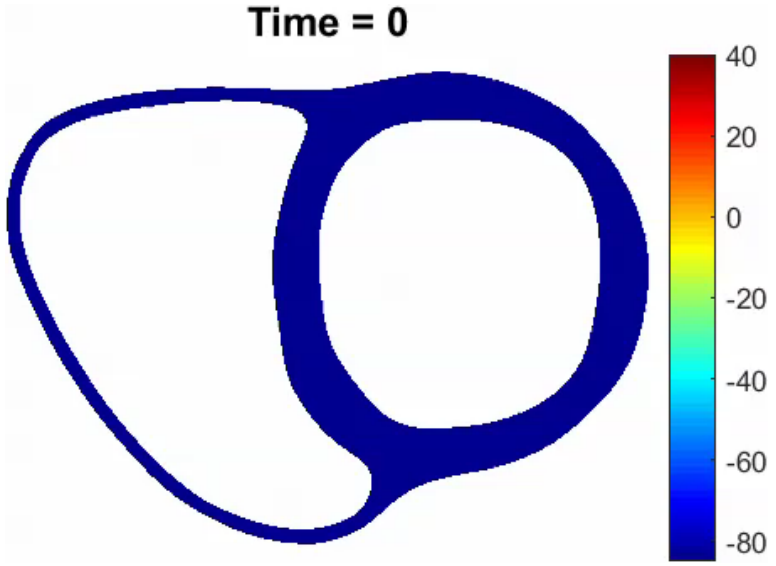
```
class TestConductivity : public CxxTest::TestSuite
{
public:
    void TestConductivity2d() throw(Exception)
    {
        // MESH
        std::string filepath = "projects/anamin/test/data/2Dmesh";
        DistributedTetrahedralMesh<2,2> mesh;
        TrianglesMeshReader<2,2> mesh_reader(filepath);
        mesh.ConstructFromMeshReader(mesh_reader);

        // NUMERICS
        double odeT, pdeT, Print_Time;
        odeT =  0.025;
        pdeT =  0.05;
        Print_Time=1;  // ms

        HeartConfig::Instance()->SetOdePdeAndPrintingTimeSteps(odeT, pdeT, Print_Time);

        // BIDOMAIN PARAMETERS
        HeartConfig::Instance()->SetSurfaceAreaToVolumeRatio(1400);     // 1/cm
        HeartConfig::Instance()->SetCapacitance(1.0);                   // uF/cm^2
        HeartConfig::Instance()->SetIntracellularConductivities(Create_c_vector(0.1, 0.1));
        HeartConfig::Instance()->SetExtracellularConductivities(Create_c_vector(0.4, 0.4));

        // SIMULATION TIME
        HeartConfig::Instance()->SetSimulationDuration(1200); //ms

        //POSTPROCESSING OPTIONS
        // APD map
        std::vector<std::pair<double,double> > apd_map;
        apd_map.push_back(std::pair<double, double>(90.0, 0.0));    // APD90, 0 mV threshold
        HeartConfig::Instance()->SetApdMaps(apd_map);
```

```
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"

class MyCellFactory : public AbstractCardiacCellFac
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus;

public:
    MyCellFactory(float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0,
    {
    }

    AbstractCvodeCell* CreateCardiacCellForTissueNo
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_c

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) &&
        {
            p_cell = new CellORd2011epi_fkatpFromCe

        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFromCe
        }
        return p_cell;
    }
};

class TestConductivity : public CxxTest::TestSuite
{
public:
    void TestConductivity2d() throw(Exception)
    {
        // MESH
        std::string filepath = "projects/anamin/tes
        DistributedTetrahedralMesh<2,2> mesh;
        TrianglesMeshReader<2,2> mesh_reader(filepa
        mesh.ConstructFromMeshReader(mesh_reader);

        // NUMERICS
        double odeT, pdeT, Print_Time;
        odeT =  0.025;
        pdeT =  0.05;
        Print_Time=1;  // ms

        HeartConfig::Instance()->SetOdePdeAndPrinti

        // BIDOMAIN PARAMETERS
        HeartConfig::Instance()->SetSurfaceAreaToVo
        HeartConfig::Instance()->SetCapacitance(1.0
        HeartConfig::Instance()->SetIntracellularCo
        HeartConfig::Instance()->SetExtracellularCo

        // SIMULATION TIME
        HeartConfig::Instance()->SetSimulationDurat

        //POSTPROCESSING OPTIONS
        // APD map
        std::vector<std::pair<double,double> > apd_
        apd_map.push_back(std::pair<double, double>
        HeartConfig::Instance()->SetApdMaps(apd_map

        // Activation time map
        std::vector<double> upstroke_time_map;
        upstroke_time_map.push_back(0.0);
        HeartConfig::Instance()->SetUpstrokeTimeMap

        // Max Upstroke Velocity map
        std::vector<double> maxupstroke_vel_map;
        maxupstroke_vel_map.push_back(0.0);
        HeartConfig::Instance()->SetMaxUpstrokeVelo


        // OUTPUT OPTIONS
        HeartConfig::Instance()->SetOutputDirectory
        HeartConfig::Instance()->SetOutputFilenameP
        HeartConfig::Instance()->SetOutputUsingOrig
        HeartConfig::Instance()->SetVisualizeWithMe
        HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

        // PARAMETERS
        double period           = 600.0;

        MyCellFactory cell_factory(period);

        // PROBLEM
        BidomainProblem<2> bidomain_problem( &cell_factory );

        bidomain_problem.SetMesh(&mesh);
        bidomain_problem.SetWriteInfo();
        bidomain_problem.Initialise();
        bidomain_problem.Solve();
    }
};
```

```
    //POSTPROCESSING OPTIONS
    // APD map
    std::vector<std::pair<double,double> > apd_map;
    apd_map.push_back(std::pair<double, double>(90.0, 0.0));    // APD90, 0 mV threshold
    HeartConfig::Instance()->SetApdMaps(apd_map);

    // Activation time map
    std::vector<double> upstroke_time_map;
    upstroke_time_map.push_back(0.0);                           // 0 mV threshold
    HeartConfig::Instance()->SetUpstrokeTimeMaps(upstroke_time_map);

    // Max Upstroke Velocity map
    std::vector<double> maxupstroke_vel_map;
    maxupstroke_vel_map.push_back(0.0);
    HeartConfig::Instance()->SetMaxUpstrokeVelocityMaps(maxupstroke_vel_map);

    // OUTPUT OPTIONS
    HeartConfig::Instance()->SetOutputDirectory("./OUT");
    HeartConfig::Instance()->SetOutputFilenamePrefix("results");
    HeartConfig::Instance()->SetOutputUsingOriginalNodeOrdering(true);
    HeartConfig::Instance()->SetVisualizeWithMeshalyzer(false);
    HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

    // PARAMETERS
    double period           = 600.0;
    MyCellFactory cell_factory(period);

    // PROBLEM
    BidomainProblem<2> bidomain_problem( &cell_factory );

    bidomain_problem.SetMesh(&mesh);
    bidomain_problem.SetWriteInfo();
    bidomain_problem.Initialise();
    bidomain_problem.Solve();
}
```



- ✓ Similar APD values in all the geometry
- ✓ Activation time map starting from the septum
- ✓ Similar maximum upstroke velocities

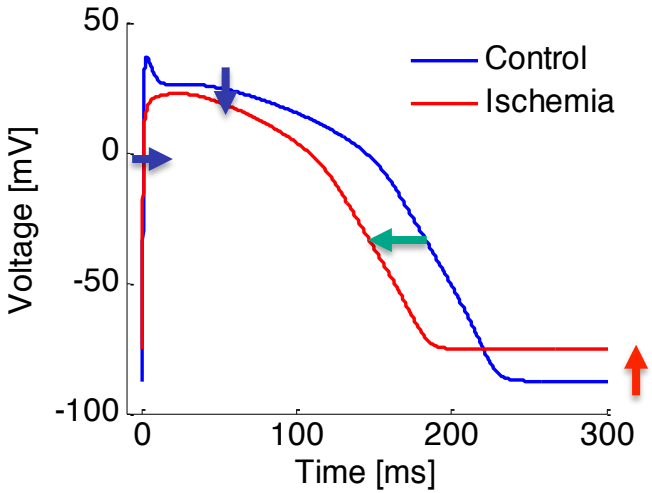


Chaste example 2

- 2D propagation with ischemia

Ischemia in the anterior heart region:

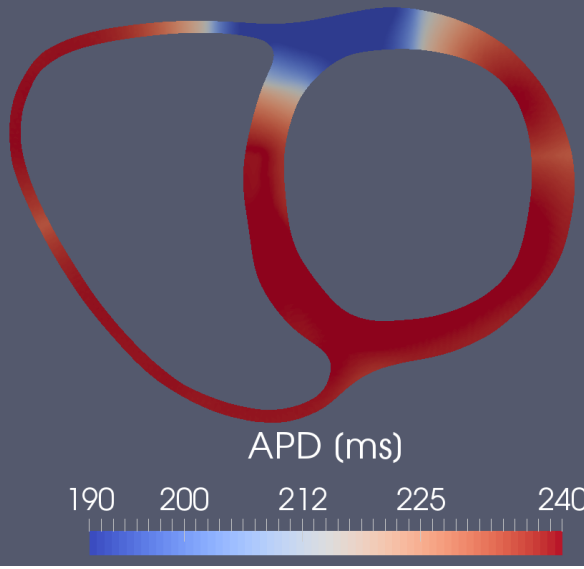
- Hyperkalaemia: [K]o = 8.5 mM
- Hypoxia: fkatp = 0.04
- Acidosis: ↓ 25% g_{Na} and g_{CaL}

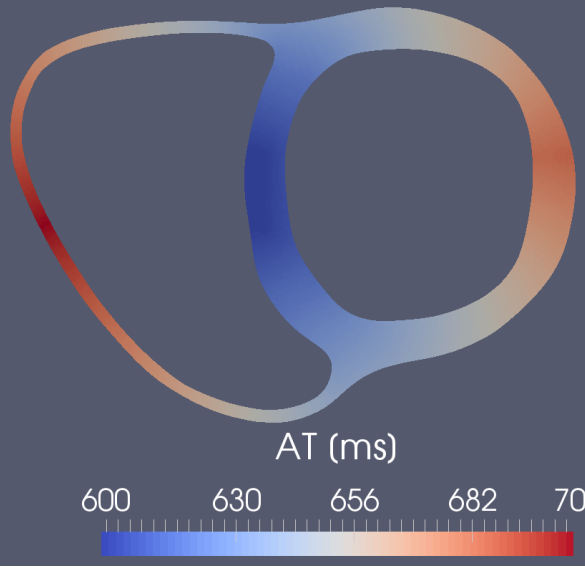


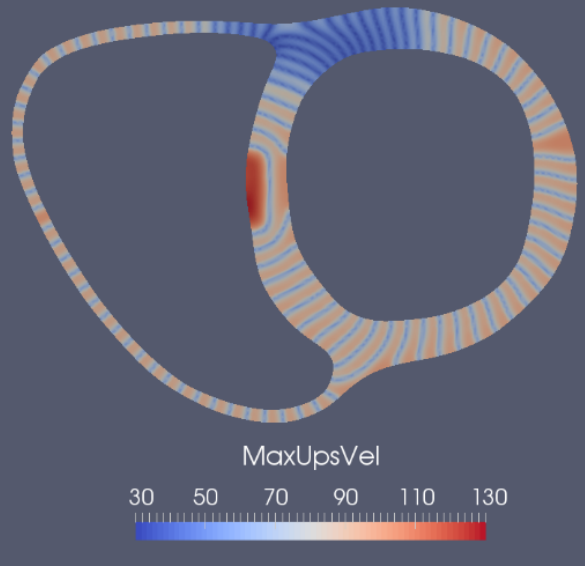
{Dutta et al, PBMB 2017}

```
class MyCellFactory : public AbstractCardiacCellFactory<2>
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus;
    double k;
    double fkatp;
    double sf_na_ca;

public:
    MyCellFactory(float Input_Ko, float Input_fkatp, float SF_Na_Ca, float period)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0, 1, period, 0))
    {
        k = Input_Ko;
        fkatp = Input_fkatp;
        sf_na_ca = SF_Na_Ca;
    }

    AbstractCvodeCell* CreateCardiacCellForTissueNode(Node<2>* pNode)
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_cell;

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) && (y<0.1) )
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpStimulus);

        }
        else if ((x>-0.4) && (x<0.05) && (y>0.25) && (y<0.44))
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
            p_cell->SetParameter("extracellular_potassium_concentration",k);
            double gna = p_cell->GetParameter("membrane_fast_sodium_current_conductance");
            double gca = p_cell->GetParameter("membrane_L_type_calcium_current_conductance");
            p_cell->SetParameter("membrane_fast_sodium_current_conductance",gna*sf_na_ca);
            p_cell->SetParameter("membrane_L_type_calcium_current_conductance",gca*sf_na_ca);
            p_cell->SetParameter("membrane_atp_dependent_potassium_current_conductance",fkatp);
        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
        }
```

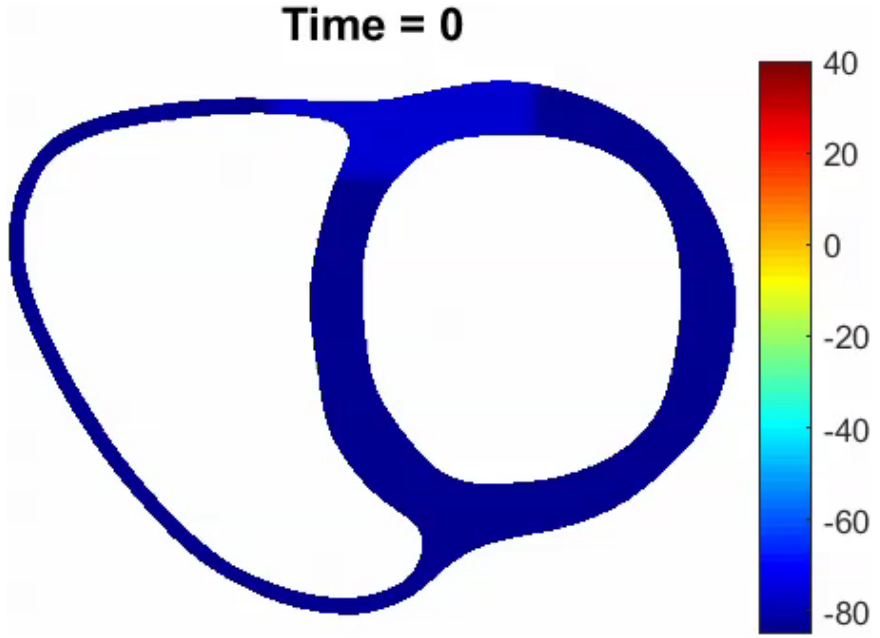
```
    //POSTPROCESSING OPTIONS
    // APD map
    std::vector<std::pair<double,double> > apd_map;
    apd_map.push_back(std::pair<double, double>(90.0, 0.0));    // APD90, 0 mV threshold
    HeartConfig::Instance()->SetApdMaps(apd_map);

    // Activation time map
    std::vector<double> upstroke_time_map;
    upstroke_time_map.push_back(0.0);                           // 0 mV threshold
    HeartConfig::Instance()->SetUpstrokeTimeMaps(upstroke_time_map);

    // Max Upstroke Velocity map
    std::vector<double> maxupstroke_vel_map;
    maxupstroke_vel_map.push_back(0.0);
    HeartConfig::Instance()->SetMaxUpstrokeVelocityMaps(maxupstroke_vel_map);


    // OUTPUT OPTIONS
    HeartConfig::Instance()->SetOutputDirectory("./OUT");
    HeartConfig::Instance()->SetOutputFilenamePrefix("results");
    HeartConfig::Instance()->SetOutputUsingOriginalNodeOrdering(true);
    HeartConfig::Instance()->SetVisualizeWithMeshalyzer(false);
    HeartConfig::Instance()->SetVisualizeWithParallelVtk(true);

    // PARAMETERS
    double Input_Ko         = 8.5;
    double Input_fkatp      = 0.05;
    double SF_Na_Ca         = 0.75;

    double period           = 600.0;

    MyCellFactory cell_factory(Input_Ko, Input_fkatp, SF_Na_Ca, period);


    // PROBLEM
    BidomainProblem<2> bidomain_problem( &cell_factory );

    bidomain_problem.SetMesh(&mesh);
    bidomain_problem.SetWriteInfo();
    bidomain_problem.Initialise();
    bidomain_problem.Solve();
}
```





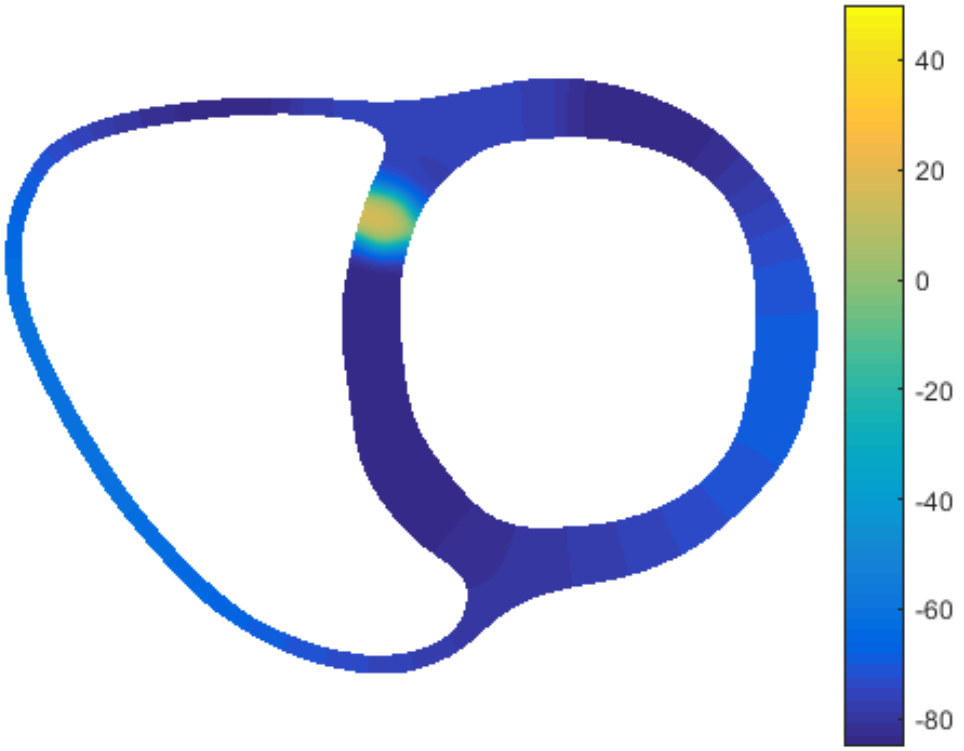


- ✓ Shorter APD values in the ischemic region
- ✓ Activation time map starting from the septum
- ✓ Maximum upstroke velocities slower in the ischemic region



Chaste example 3

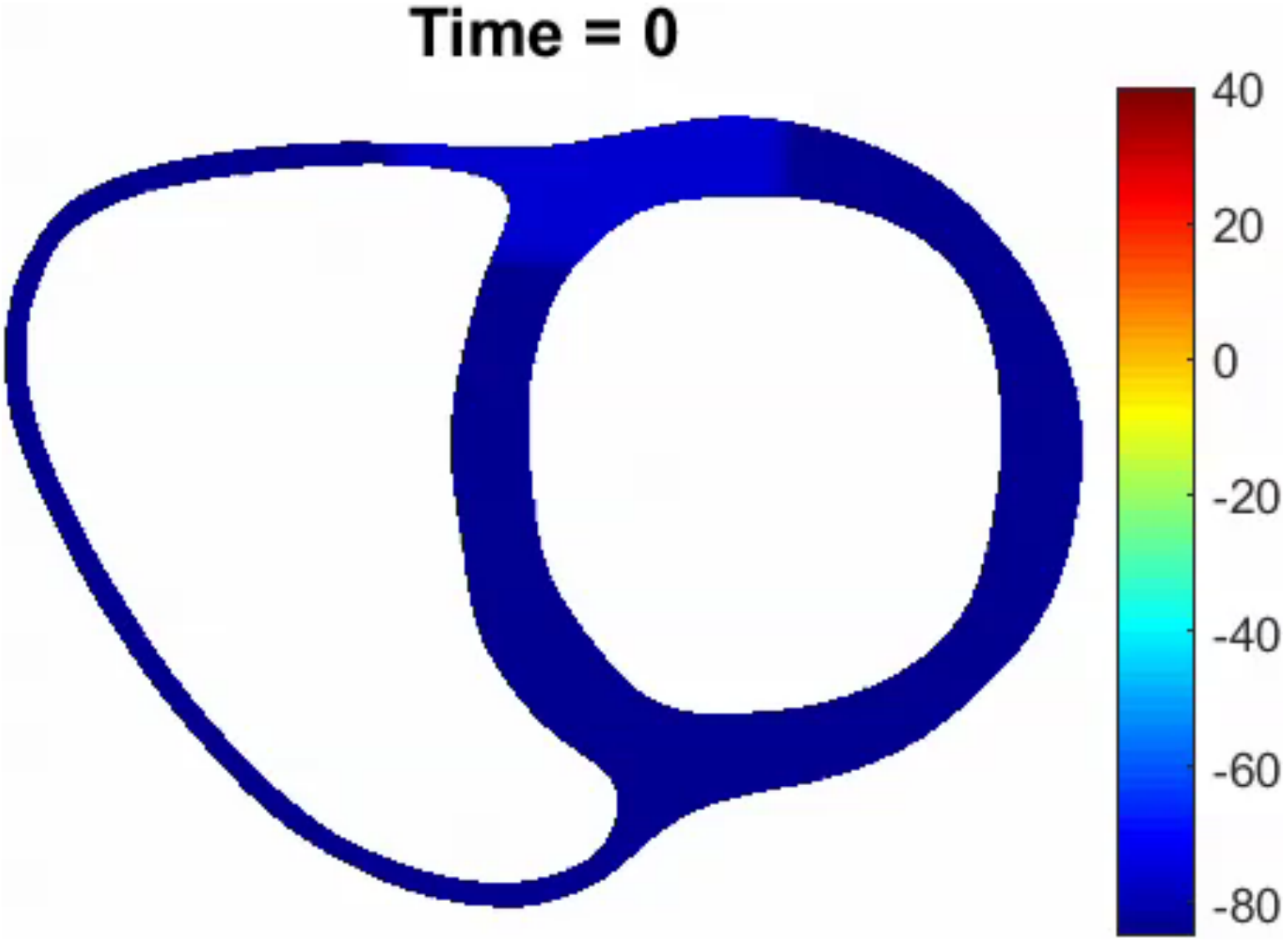
- 2D geometry with ischemia and ectopy



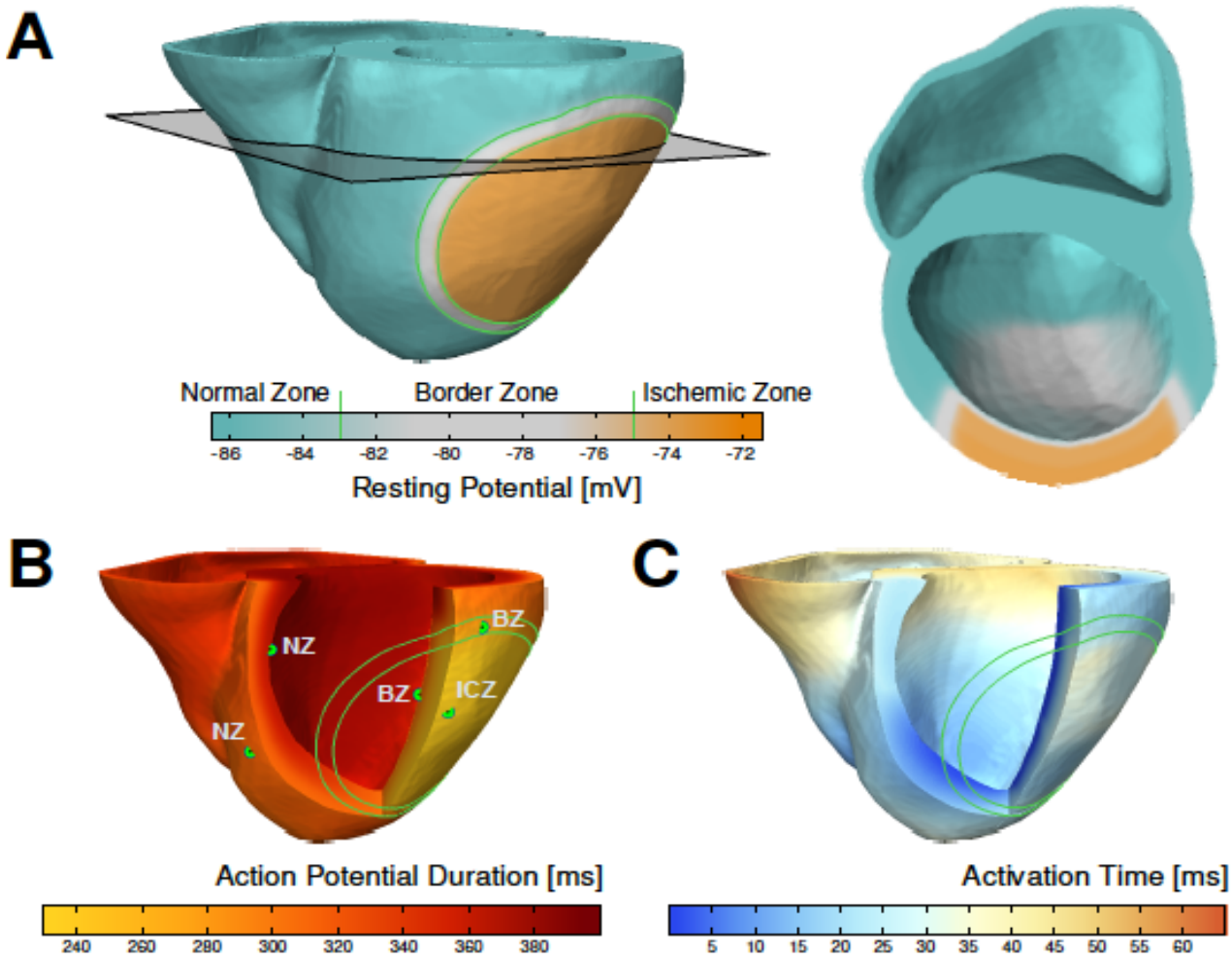
Adding the effect of an ectopic beat in the border zone region

```cpp
#include <cxxtest/TestSuite.h>
#include "PetscSetupAndFinalize.hpp"

#include "BidomainProblem.hpp"
#include "RegularStimulus.hpp"
#include "SimpleStimulus.hpp"
#include "TetrahedralMesh.hpp"

#include "ORd2011epi_fkatpCvodeOpt.hpp"

class MyCellFactory : public AbstractCardiacCellFactory<2> // <3> here
{
private:
    boost::shared_ptr<RegularStimulus> mpStimulus;
    boost::shared_ptr<SimpleStimulus> mpStimulus2;
    double k;
    double fkatp;
    double sf_na_ca;

public:
    MyCellFactory(float Input_Ko, float Input_fkatp, float SF_Na_Ca, float period, float ectopic_time)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0, 1, period, 0)),
          mpStimulus2(new SimpleStimulus(-120000.0, 2.0, ectopic_time))
    {
        k = Input_Ko;
        fkatp = Input_fkatp;
        sf_na_ca = SF_Na_Ca;
    }
    AbstractCvodeCell* CreateCardiacCellForTissueNode(Node<2>* pNode)
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_cell;

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) && (y<0.1) )
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpStimulus);
        }
```

```
public:
    MyCellFactory(float Input_Ko, float Input_fkatp, float SF_Na_Ca, float period, float ectopic_time)
        : AbstractCardiacCellFactory<2>(),
          mpStimulus(new RegularStimulus(-120000.0, 1, period, 0)),
          mpStimulus2(new SimpleStimulus(-120000.0, 2.0, ectopic_time))
    {
        k = Input_Ko;
        fkatp = Input_fkatp;
        sf_na_ca = SF_Na_Ca;
    }
    AbstractCvodeCell* CreateCardiacCellForTissueNode(Node<2>* pNode)
    {
        CellORd2011epi_fkatpFromCellMLCvodeOpt* p_cell;

        double x = pNode->rGetLocation()[0];
        double y = pNode->rGetLocation()[1];

        if ( (x>-0.32) && (x<-0.27) && (y>-0.1) && (y<0.1) )
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpStimulus);
        }

        else if ((x>-0.4) && (x<0.05) && (y>0.25) && (y<0.44))
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
            p_cell->SetParameter("extracellular_potassium_concentration",k);
            double gna = p_cell->GetParameter("membrane_fast_sodium_current_conductance");
            double gca = p_cell->GetParameter("membrane_L_type_calcium_current_conductance");
            p_cell->SetParameter("membrane_fast_sodium_current_conductance",gna*sf_na_ca);
            p_cell->SetParameter("membrane_L_type_calcium_current_conductance",gca*sf_na_ca);
            p_cell->SetParameter("membrane_atp_dependent_potassium_current_conductance",fkatp);
        }
        else
        {
            p_cell = new CellORd2011epi_fkatpFromCellMLCvodeOpt(mpSolver, mpZeroStimulus);
        }

        //Definition Ectopic region
        ChastePoint<2> stim_centre (-0.43,0.39);
        ChastePoint<2> stim_radius (0.05,0.05);
        ChasteEllipsoid<2> stim_region (stim_centre,stim_radius);

        // Modification of cell properties if it is in the ectopic region
        bool cell_is_in_stim_region = stim_region.DoesContain(this->GetMesh()->GetNode(pNode->GetIndex())->rGetLocation());

        if(cell_is_in_stim_region)
        {
            p_cell->SetStimulusFunction(mpStimulus2);
        }
        return p_cell;
```


Time = 0
40
20
0
-20
-40
-60
-80

3D simulations in Chaste

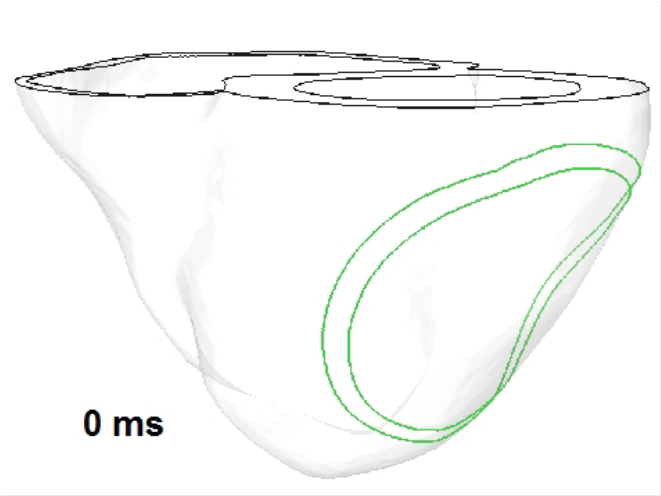


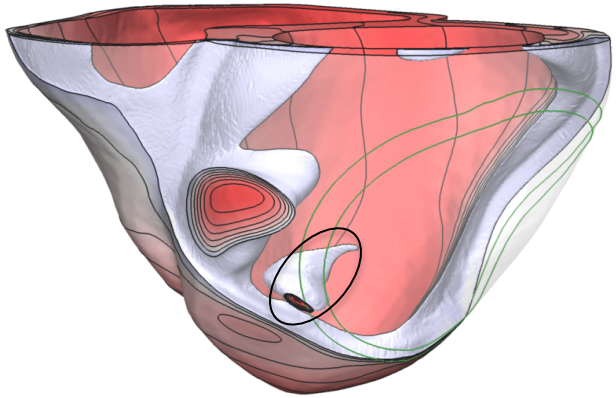
- Inclusion of border zones: endocardial and around the ischemic area
- Transmural heterogeneities

Dutta and Minchole et al, PBMB 2016

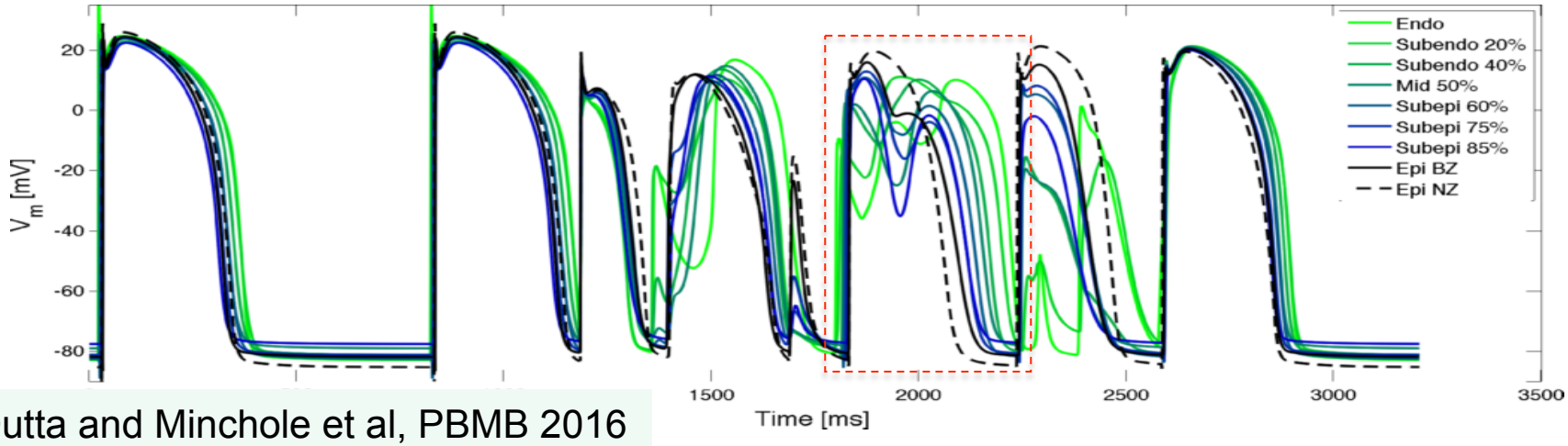
3D simulations in Chaste

Simulated effect of IKr block in acute ischaemia





1831 ms

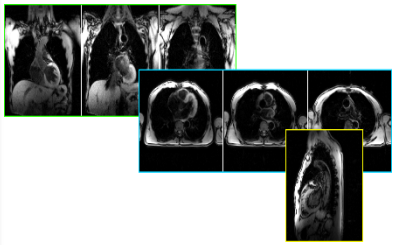


Dutta and Minchole et al, PBMB 2016

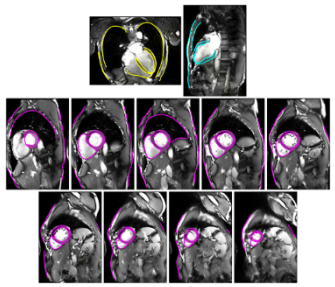
Personalization of anatomical models

MRI

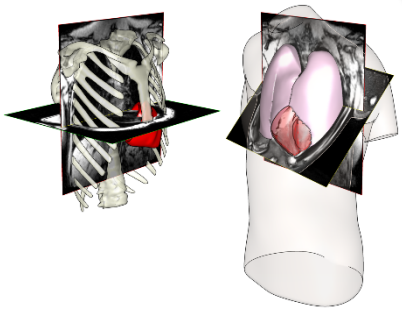
LOCALIZERS

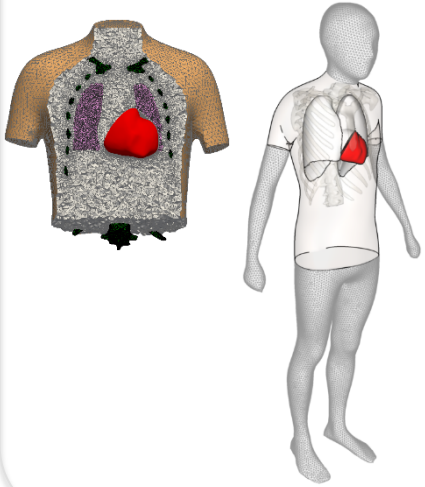


CINE

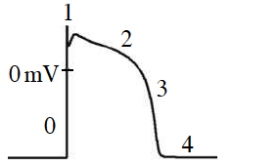


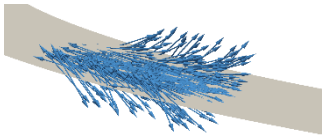
Meshes

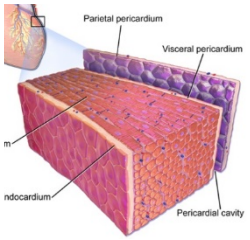


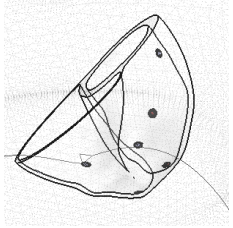


Parametrization

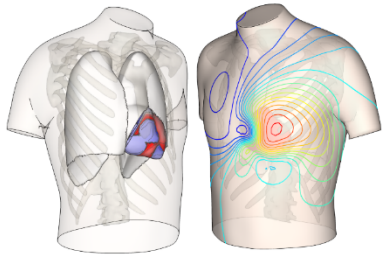




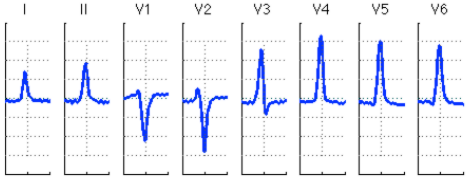




ELECTRO-PHYSIOLOGY

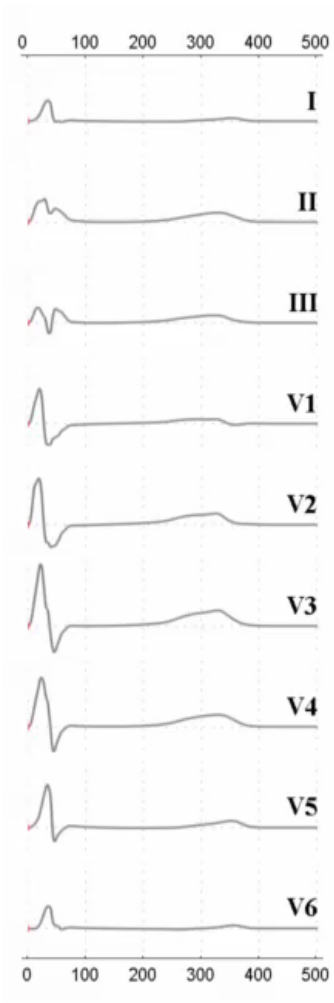


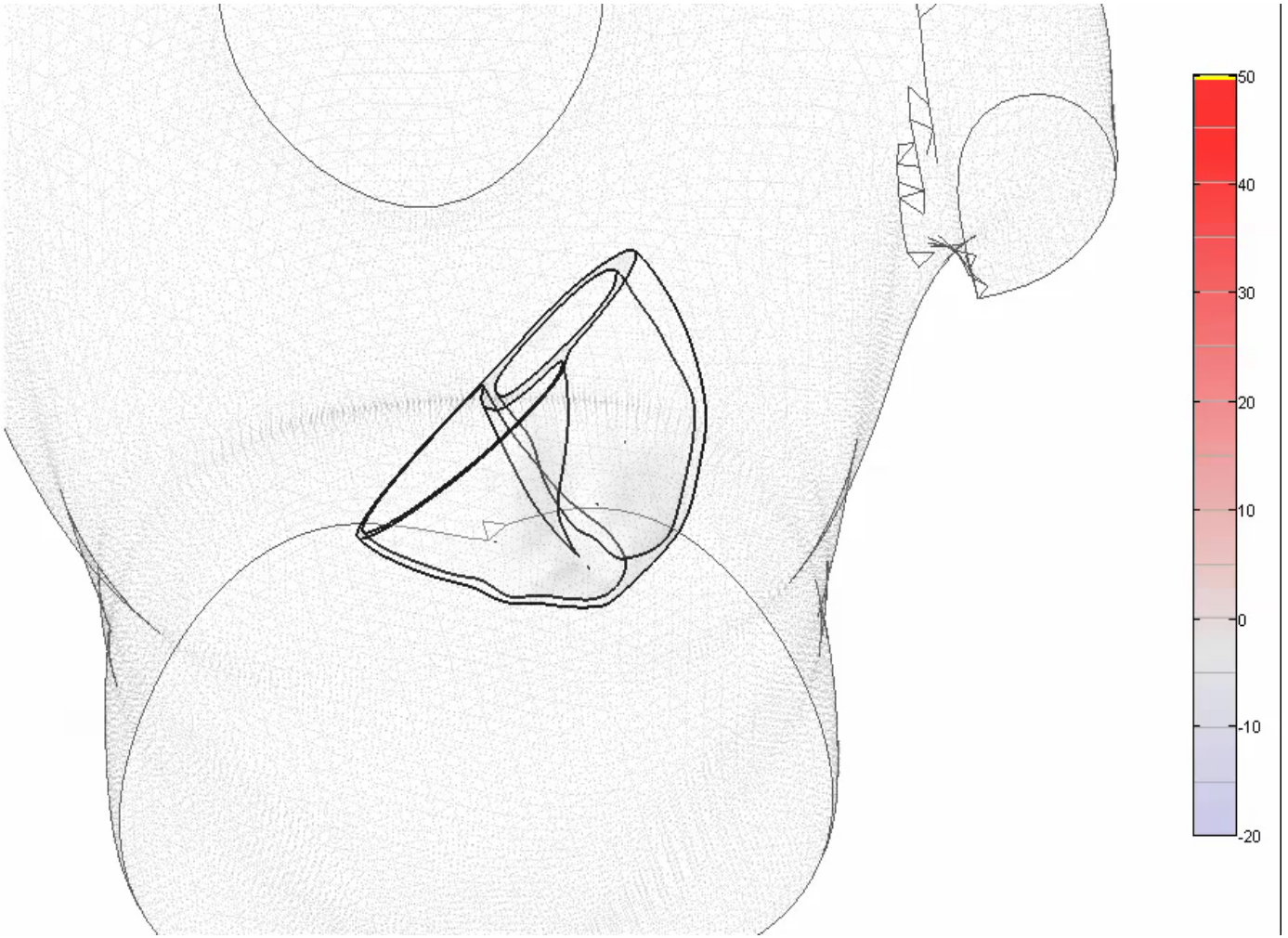
ECG



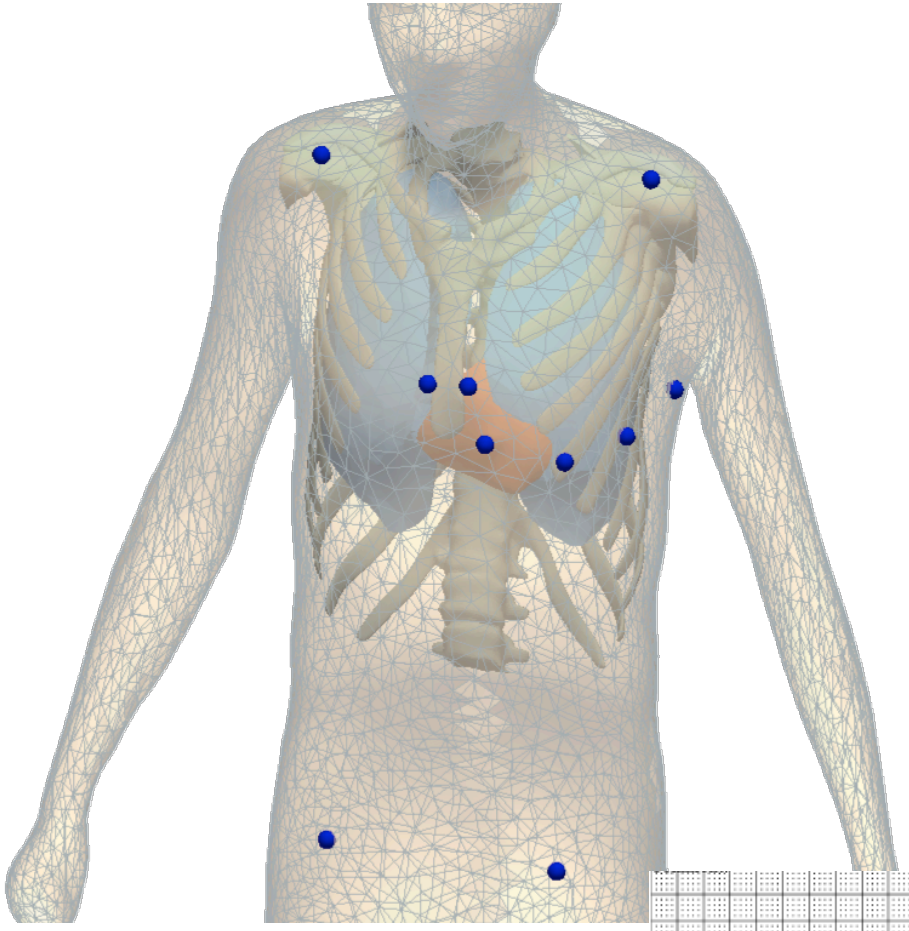
Zacur, Minchole et al, BIVPCS 2017

Subject DTI003

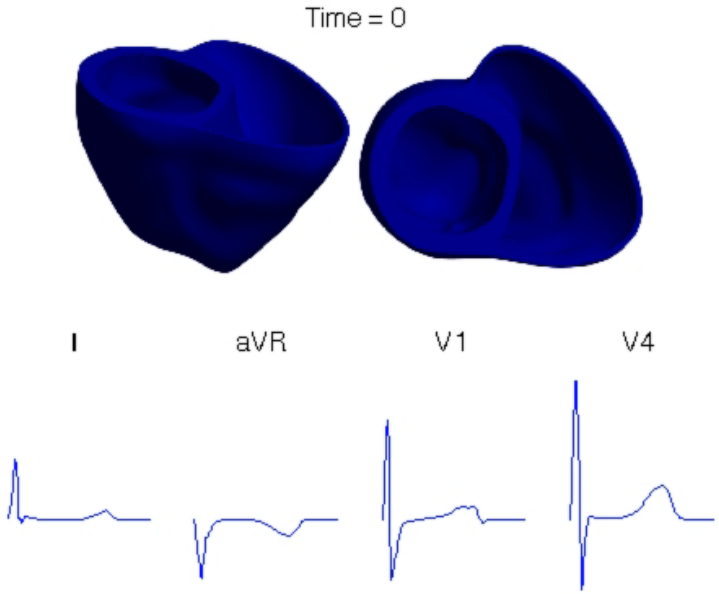


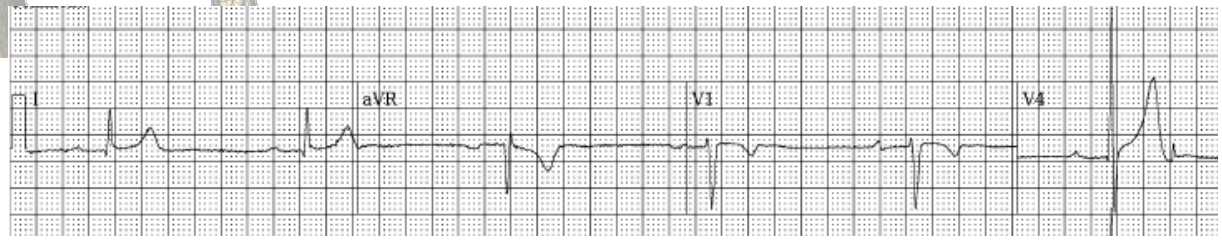


Computer Simulations to explain Cardiac phenotypes



Linking structure and function
HPC cardiac simulation





Example of electrophysiology with Alya

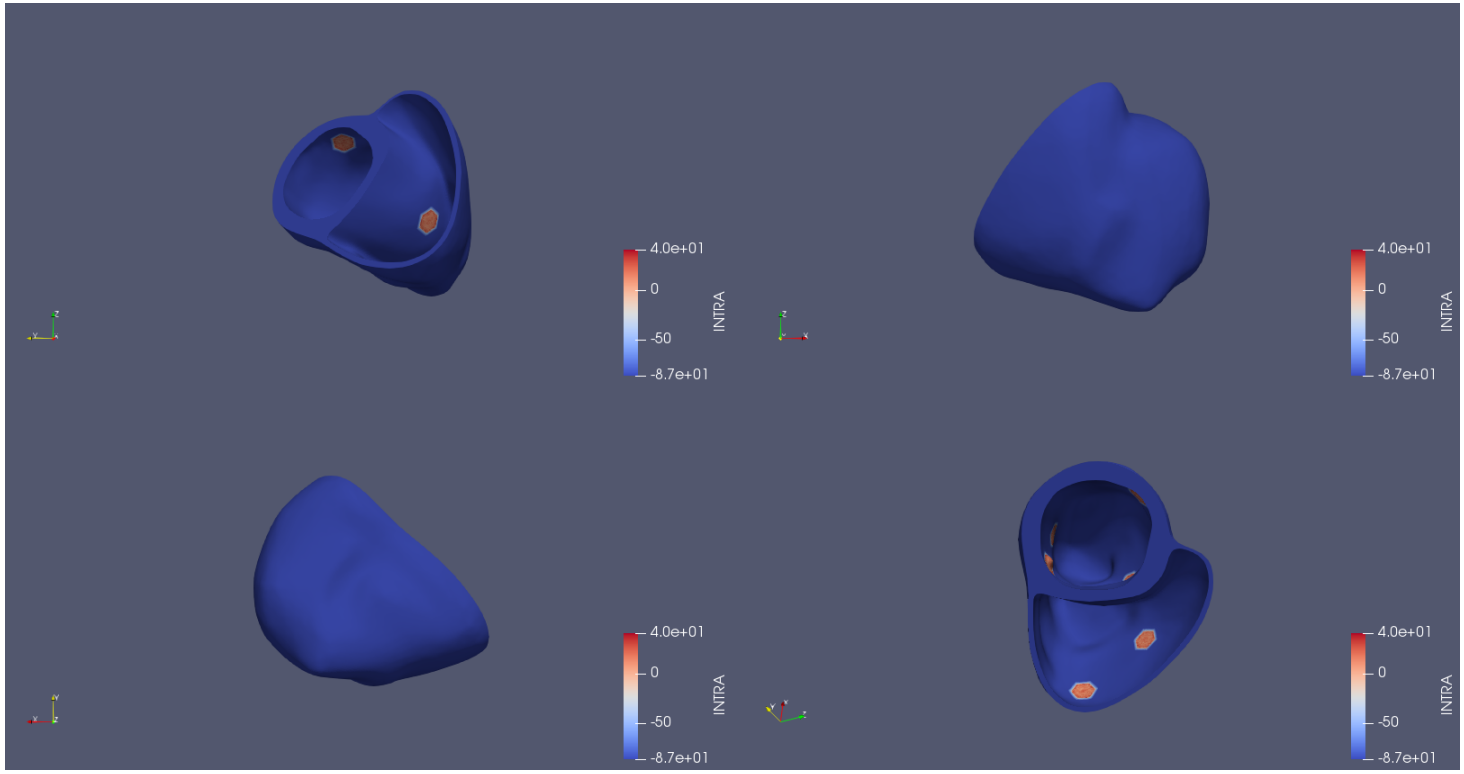
Alya

- Alya functionalities:
  - ✓ Monodomain and bidomain models
  - ✓ A few cellular action potential models are implemented (FHN, ORd, TT)
  - ✓ Bidirectional electro-mechanical coupling is already implemented and fully functional
  - ✓ Excitation-contraction coupling models and models of stretch-activated ion channels are implemented
  - ✓ High efficiency and very high scalability
  - ✓ Other physics are also available (combustion, incompressible and compressible CFD…)

To obtain Alya, contact: mariano.vazquez@bsc.es

Alya example 1

- Biventricular model (5.2M nodes and 32M elements)
- Run on MareNostrum IV (BSC) on 2,000 cores in 15 min
- This example comprises of 4 input files in ASCII format: .dat, .dom.dat, .ker.dat and .exm.dat
- Electrical propagation only



Electro-mechanical modelling

Equations of motion

$$\nabla \cdot \mathbf{P} + \rho_0 \mathbf{b} = \rho_0 \ddot{\mathbf{u}}$$

Hyperelastic model

$$\psi(\mathbf{C}) = \frac{a}{2b}\exp[b(I_1 - 3)] + \sum_{i=f,s}\{\exp[b_i(I_{4i} - 1)^2] - 1\} + \frac{a_{fs}}{2b_{fs}}[\exp(b_{fs}I_{8fs}^2) - 1]$$

Active contraction model

$$\frac{\mathrm{d}\mathbf{s}}{\mathrm{d}t} = m_s(\mathbf{c}, \mathbf{s}, \lambda, \dot{\lambda})$$

Active tension generation

$$T_{act} = h(\lambda)\frac{T_{ref}}{r_s}[(\zeta_s + 1)S + \zeta_w W]$$

Mechano-electric coupling

$$\nabla \cdot (J\mathbf{F}^{-1}\mathbf{D}_0\mathbf{F}^{-T}\nabla V) = \chi(C\frac{\mathrm{d}V}{\mathrm{d}t} + I_{ion} - I_{app})$$

Electro-mechanic coupling

$$\frac{\mathrm{d}\mathbf{s}}{\mathrm{d}t} = m_s(\mathbf{c}, \mathbf{s}, \lambda, \dot{\lambda})$$

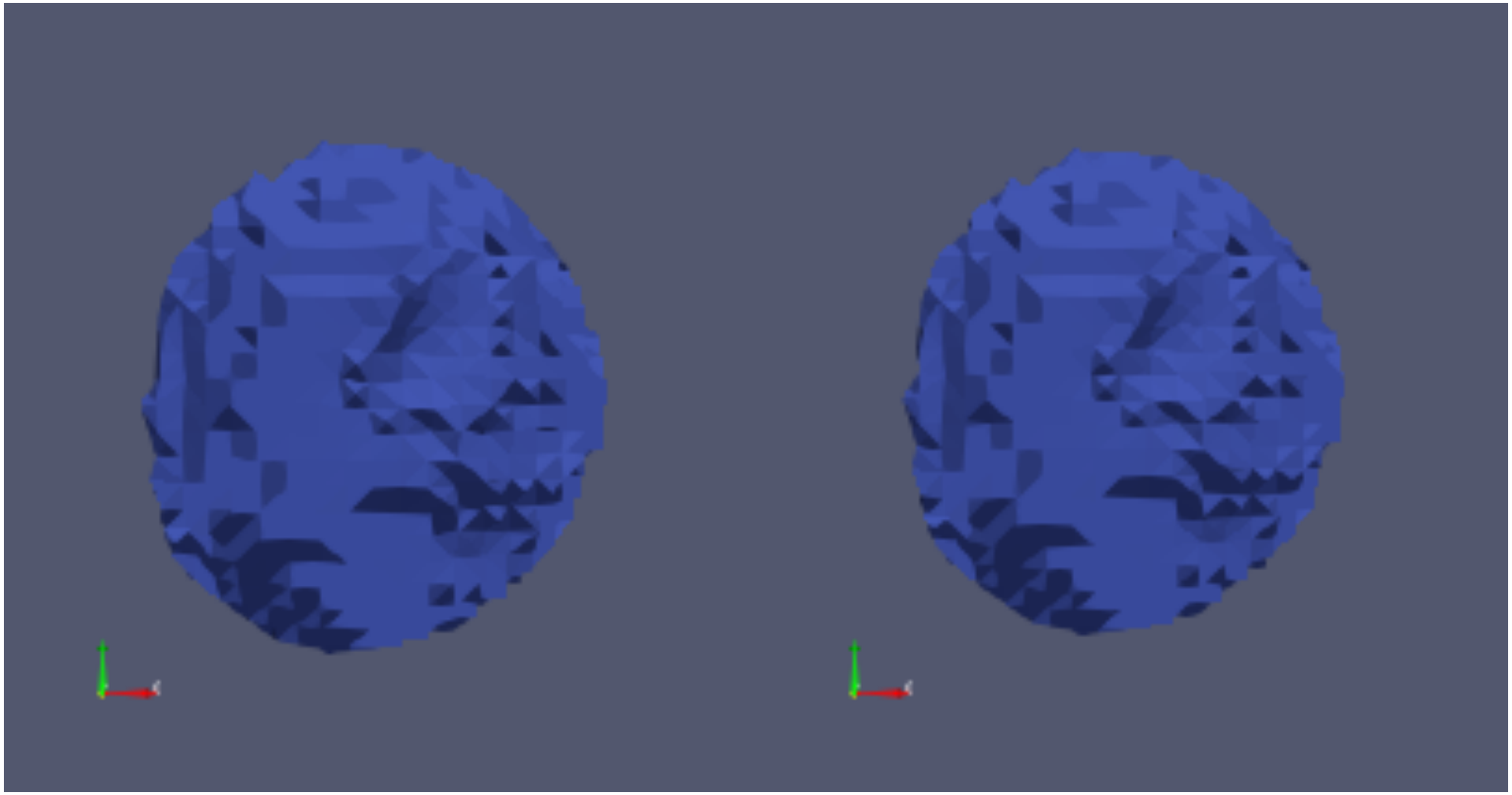
$$T_{act} = h(\lambda)\frac{T_{ref}}{r_s}[(\zeta_s + 1)S + \zeta_w W]$$

Stretch-activated ion channels

$$I_{inw}(V, \mathbf{F}) = f(\sqrt{\lambda} - 1)(V - E)$$

Alya example 2

- ✓ Small blob of tissue
- ✓ Run locally with 4 cores
- ✓ Electromechanical simulation, Hunter(left) vs Land(right)



Alya .dat and .dom.dat

```
$-----------------------------------------------------------------
RUN_DATA
  ALYA:  heart_test
END_RUN_DATA
$-----------------------------------------------------------------
PROBLEM_DATA
  MAXIMUM_NUMBER_GLOBAL:  1
  NUMBER_OF_STEPS:        1000000
  TIME_INTERVAL:          0.000, 0.400
$-----------------------------------------------------------------
  EXMEDI:           ON
  END_EXMEDI
  $SOLIDZ_PROBLEM:  ON
  $END_SOLIDZ
  PARALL_SERVICE:   ON
    OUTPUT_FILE:    NO
    POSTPROCESS:    MASTER
    PARTITION:      FACES
  END_PARALL
$-----------------------------------------------------------------
END_PROBLEM_DATA
$-----------------------------------------------------------------
```

```
$-----------------------------------------------------------
DIMENSIONS
  NODAL_POINTS =      5196552
  ELEMENTS =          31702122
  SPACE_DIMENSIONS =  3
  NODES =             4
  BOUNDARIES =        920090
END_DIMENSIONS
$-----------------------------------------------------------
STRATEGY
  INTEGRATION_RULE =           OPEN
  DOMAIN_INTEGRATION_POINTS =  1
END_STRATEGY
$-----------------------------------------------------------
GEOMETRY, GID, WALL_DISTANCE = 0.0, ROUGHNESS = 0.0
  FIELDS, NUMBER = 3
    FIELD = 1, DIMENSION = 3, NODES
      INCLUDE  heart_test.fiber
    END_FIELD
    FIELD = 2, DIMENSION = 1, NODES
      INCLUDE  heart_test.celltype
    END_FIELD
    INCLUDE heart_test.stimuli
  END_FIELDS
  INCLUDE  heart_test.geo
  INCLUDE  heart_test.surfaces
END_GEOMETRY
$-----------------------------------------------------------
BOUNDARIES, EXTRAPOLATE
$  INCLUDE heart_test.boundaries
END_BOUNDARIES
```

{http://bsccase02.bsc.es/alya/}

Alya .ker.dat and .exm.dat

```
$-----------------------------------------------------------------
  PHYSICAL_PROBLEM
    $COUPLING
    $    SOLIDZ EXMEDI
    $END_COUPLING
  END_PHYSICAL_PROBLEM
$-----------------------------------------------------------------
  NUMERICAL_TREATMENT
     MESH
$       DIVISION = 1
     END_MESH
  END_NUMERICAL_TREATMENT
$-----------------------------------------------------------------
  OUTPUT_&_POST_PROCESS
    ON_LAST_MESH
    STEPS = 50
  END_OUTPUT_&_POST_PROCESS
$-----------------------------------------------------------------
```

```
$-----------------------------------------------------------
PHYSICAL_PROBLEM
$-----------------------------------------------------------
  PROBLEM_DEFINITION
    STARTING_POTENTIAL, FIELD
     CURDENSITY: FIELD = 3
    END_STARTING_POTENTIAL
    $GEO_COUPLING: LAGRANGIAN
  END_PROBLEM_DEFINITION
$-----------------------------------------------------------
  PROPERTIES
    MATERIAL = 1
    CONTINUUM_MODEL
      IN_DIFFUSION = 0.00107143  0.00024107  0.00024107  [cm2/ms]
      INITIAL_MEMBRANE_POTENTIAL= -87.0  [mV]
    END_CONTINNUM_MODEL
    CELL_MODEL:  OHARA
      CM_CONST = 1.0
    ENDCELL_MODEL
    INI_CELL, HETEROGENEOUS
      INCLUDE heart_test.framework
    END_INI_CELL
  END_PROPERTIES
  $-----------------------------------------------------------
  FIBER_MODEL:  FIELD = 1
  CELL_TYPE:    FIELD = 2
$-----------------------------------------------------------
END_PHYSICAL_PROBLEM
$-----------------------------------------------------------
NUMERICAL_TREATMENT
  $CONSTANT_MATRIX
  SAFETY_FACTOR =          25.0
  TIMETREATMENT: IMPLICIT, CRANK
  ALGEBRAIC_SOLVER
    SOLVER:                CG
    CONVERGENCE:           ITERA=1000, TOLER=1.0e-6, ADAPTIVE, RATIO=0.01
    OUTPUT:                CONVERGENCE
    PRECONDITIONER:        DIAGONAL
  END_ALGEBRAIC_SOLVER
END_NUMERICAL_TREATMENT
$-----------------------------------------------------------
OUTPUT_&_POST_PROCESS
  POSTPROCESS INTRA
END_OUTPUT_&_POST-PROCESS
$-----------------------------------------------------------
```

{http://bsccase02.bsc.es/alya/}

Final comments

- Complex nature of biology:
  - Models as tools to augment experimental/clinical findings.
- HPC is needed for whole biventricular and torso simulations:
  - Models can several tens of millions of elements!
- Highlight of the potential of mathematical modelling for *in silico* clinical and drug trials:
  - Substitution of current protocols for human *in silico* protocols
- Highlight of the potential of mathematical modelling for explaining the underlying mechanisms of abnormalities in the ECG:
  - Potential improvement of the associated inverse problem

Acknowledgements

Computational Cardiovascular Science Group, Oxford

Ernesto Zacur, Héctor Martínez, Aurore Lyon, Alfonso Bueno, Patricia Benito, Oliver Britton, Kevin Burrage, Peter Marinov, Adam McCarthy, Polina Mamoshina, Cristian Trovato, Jakub Tomek, Francesca Margara, Julia Camps, Louie Cardone-Noott, Vicente Grau, Elisa Passini, Xin Zhou, Blanca Rodriguez

Barcelona Supercomputing Center, Spain

Jazmin Aguado-Sierra, Alfonso Santiago, Marina Lopez, Mariano Vazquez

Cardiovascular Medicine, Oxford

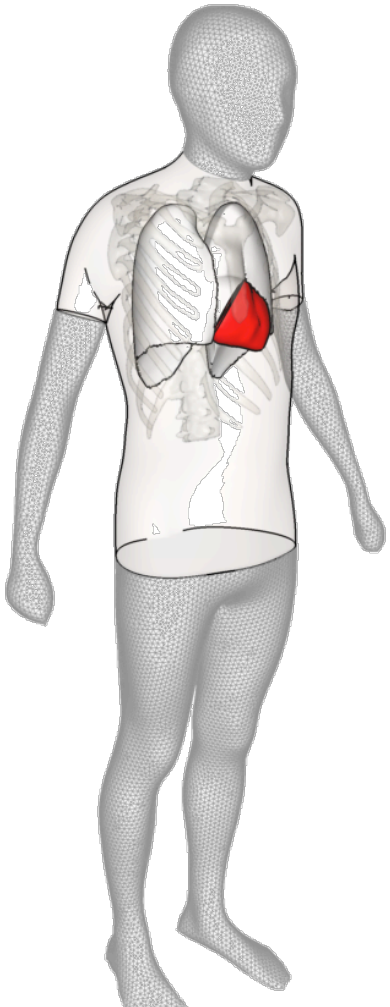
Rina Ariga, Hugh Watkins

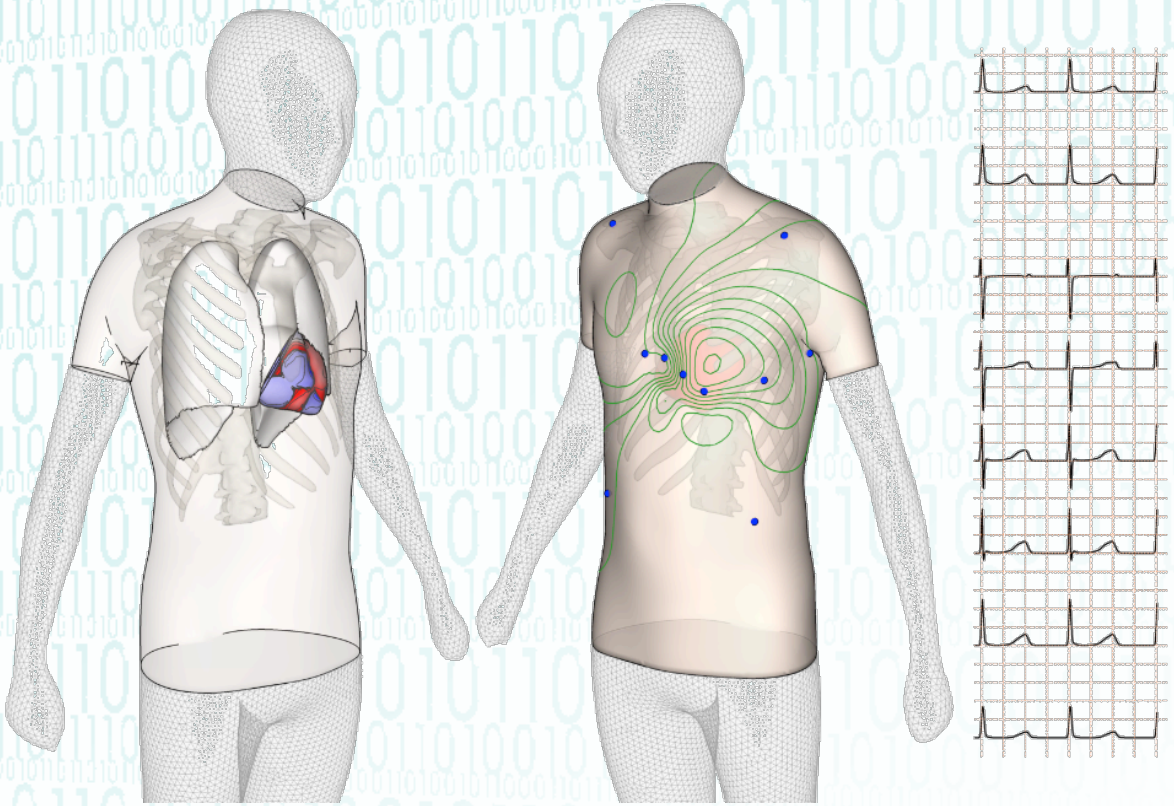
Food and Drug Administration, USA

Sara Dutta, David Strauss

Simula, Norway

Valentina Carapella





Thank You!

NC 3Rs National Centre for the Replacement Refinement & Reduction of Animals in Research
wellcome trust
British Heart Foundation
EPSRC
MRC Medical Research Council
European Commission