openCMISS & CellML/FieldML

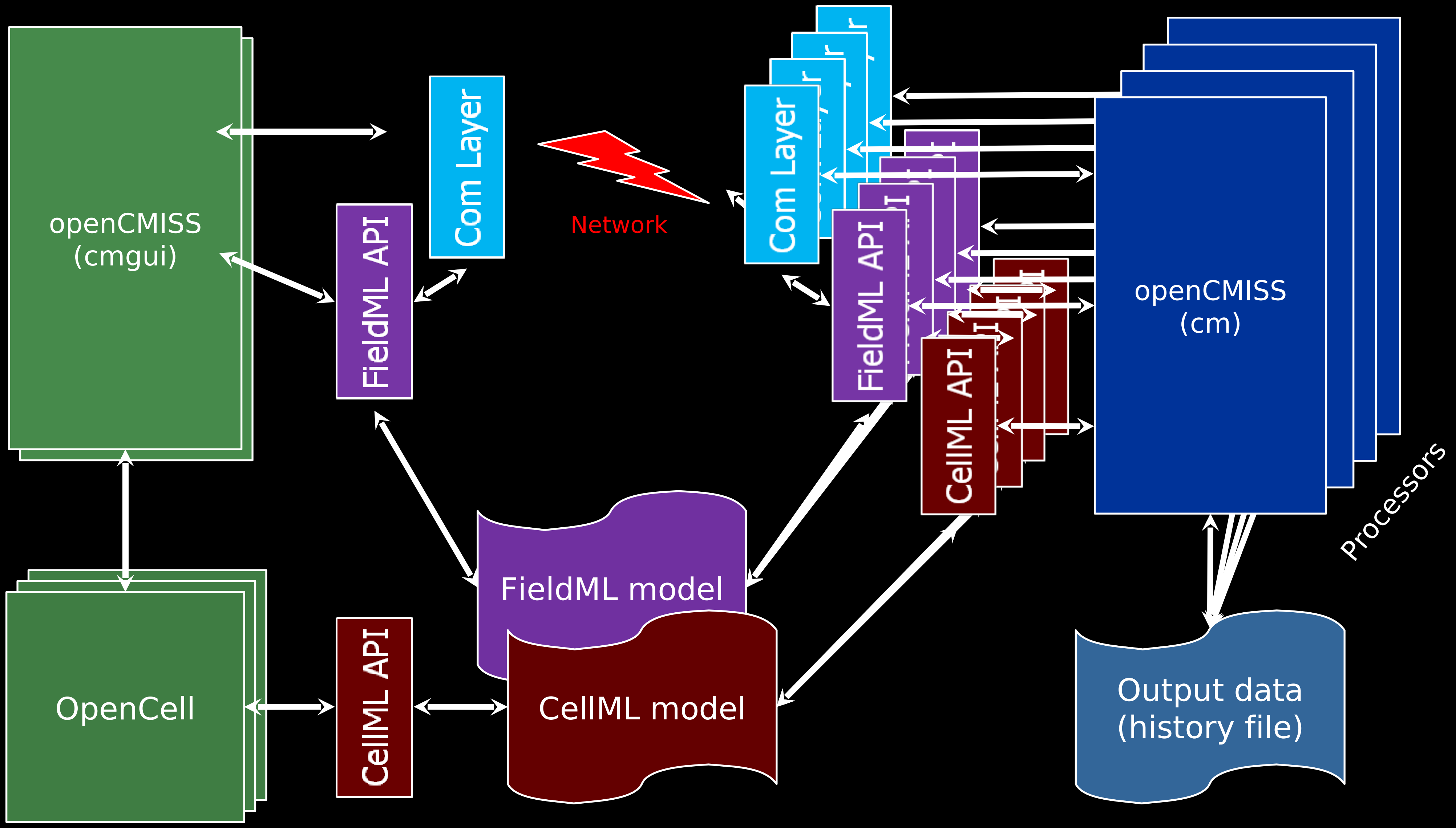
David Nickerson[1] and Chris Bradley[2]

[1]Division of Bioengineering, National University of Singapore
[2]Department of Physiology, Anatomy and Genetics, University of Oxford

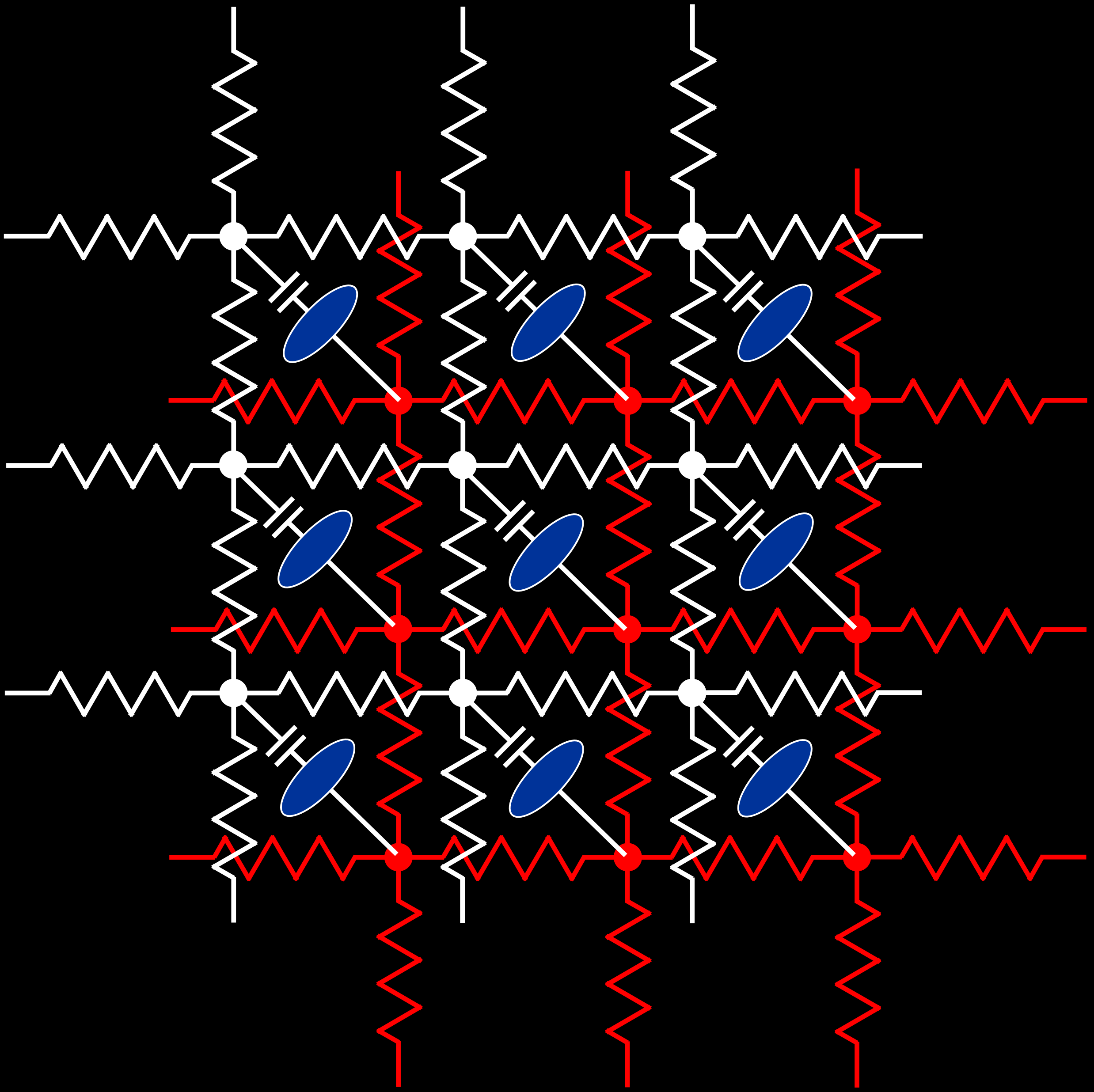
openCMISS

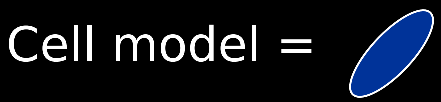
- Re-engineering of the CMISS computational engine.
- Library based approach to enable use in multiple applications.
- MPI based distributed memory + OpenMP + serial code.
- Open source on SourceForge.net
- Fortran 95 (+2003 in places).
- Object “based”.
- Now a “teenager” – about 120,000 lines.
- http://www.opencmiss.org/

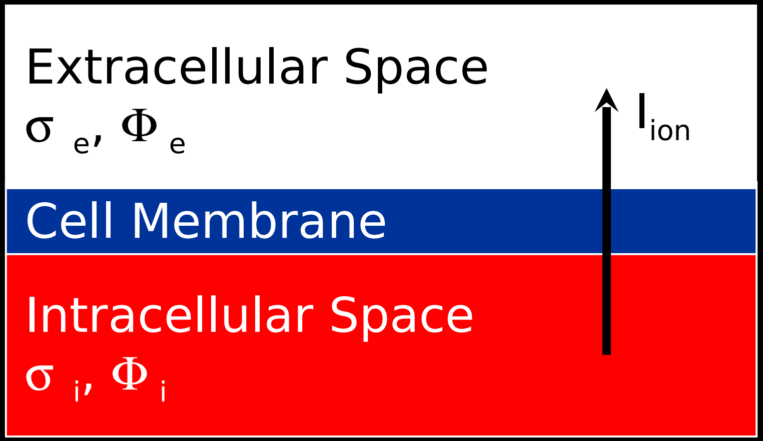
openCMISS application



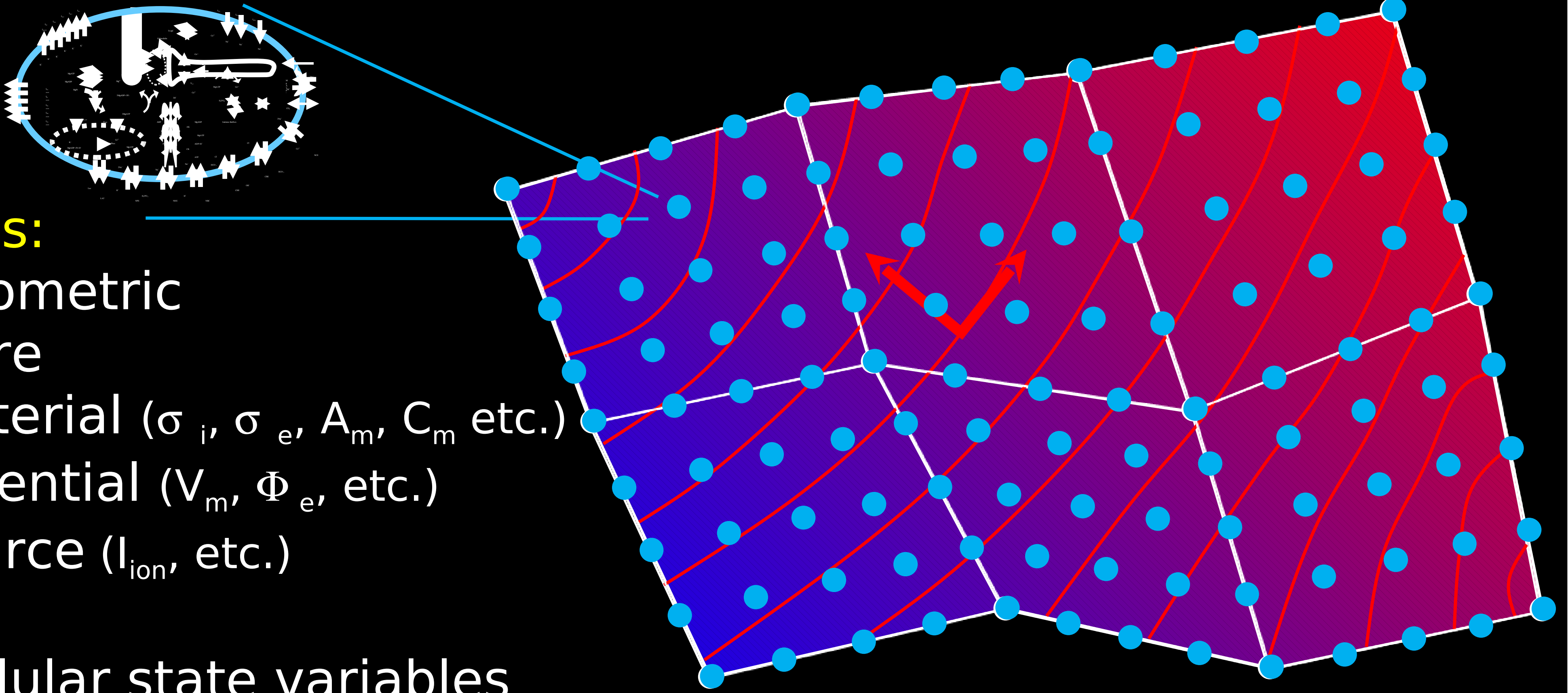
Bidomain representation







Problem/Field description



Fields:

- Geometric
- Fibre
- Material (σ_i, σ_e, A_m, C_m etc.)
- Potential (V_m, Φ_e, etc.)
- Source (I_{ion}, etc.)

- Cellular state variables
- Cellular intermediate variables
- Cellular parameter variables

Cellular parameter field

- Want to be able to vary any cell parameter spatially
- Two options
  - Interpolate every parameter before evaluating a cell model.
    - Most general
    - Computationally expensive as most parameters are constant
  - Only allow certain parameters to vary and interpolate them

Fields Involved

- Geometric field
- Fibre field
- Material fields (σ_i, σ_e, A_m, C_m etc.)
- Potential fields (V_m, Φ_e, etc.)
- Source (I_{ion}, etc.)

- Cellular state variables
- Cellular intermediate variables
- Varying cell parameters field
- Constant cell parameters field

PDE Problem

ODE Problem

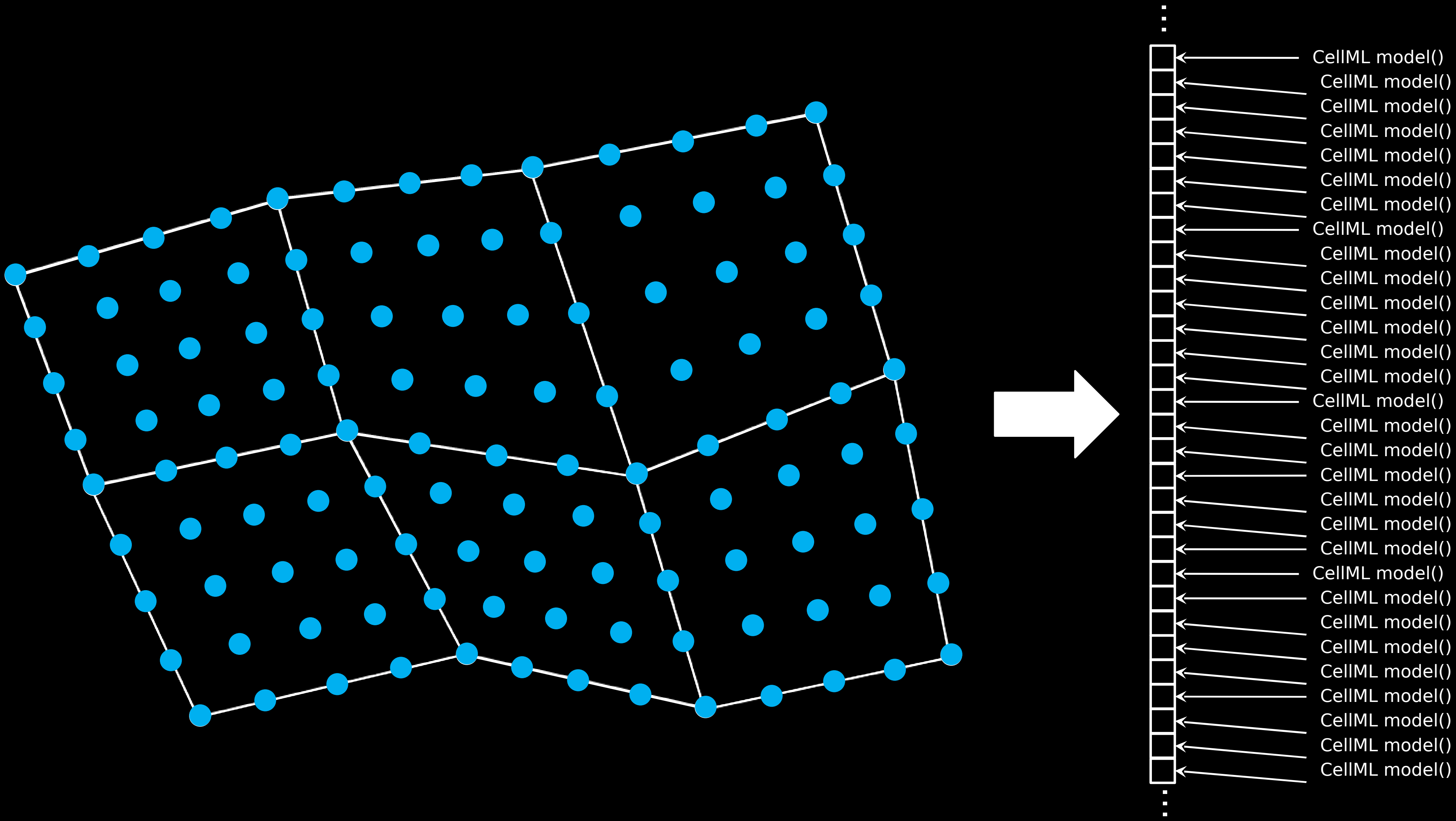
FieldML

CellML

FieldML and CellML

- FieldML is a declarative language for building hierarchical models represented by generalised mathematical fields.
  - Fields exist in $\geq$ 1 dimension
- The purpose of CellML is to store and exchange computer-based mathematical models.
  - Models are generally point based or “0D”.
- Integrate the two in openCMISS by attaching a CellML environment to a field to allow for a general way of providing a value at a field degree of freedom via an arbitrary CellML model (or function).
- Also allow interpolated fields to provide the values for CellML parameters.

Attaching CellML to a Field



Source Field

Source Field
DOF parameters

Attaching CellML to a field

- Need to specify:
  - What CellML model is associated with each of the host field degrees-of-freedom.
  - What CellML variable is returned/mapped into the field degree-of-freedom value.

- For spatially varying CellML parameters:
  - Need to specify what parameters vary
  - Need to specify what field component is mapped to what CellML variable.

Thank you!

- Chris Bradley <chris.bradley@dpag.ox.ac.uk>

CellML field interface

```
!Create the equations set source field
CALL EQUATIONS_SET_SOURCE_CREATE_START(SOURCE_FIELD_USER_NUMBER,EQUATIONS_SET, &
   & SOURCE_FIELD,ERR,ERROR,*999)
CALL EQUATIONS_SET_SOURCE_CREATE_FINISH(EQUATIONS_SET,ERR,ERROR,*999)

!Setup the CellML environment on the host source field
CALL CELLML_CREATE_START(CELLML_USER_NUMBER,SOURCE_FIELD,CELLML,ERR,ERROR,*999)
!Finish creating CellML environment
CALL CELLML_CREATE_FINISH(CELLML,ERR,ERROR,*999)

!Setup the CellML models
CALL CELLML_MODELS_CREATE_START(CELLML,ERR,ERROR,*999)
!Import the CellML models
CALL CELLML_MODELS_IMPORT(MODEL_USER_NUMBER,CELLML,URI,ERR,ERROR,*999)
!Finish creating the models.
CALL CELLML_MODELS_CREATE_FINISH(CELLML,ERR,ERROR,*999)

!Setup the CellML model field i.e., the model variable number which is used to
!evaluate the value at each of the host field dofs.
CALL CELLML_MODELS_FIELD_CREATE_START(MODEL_FIELD_USER_NUMBER,CELLML, &
   & CELLML_MODELS_FIELD,ERR,ERROR,*999)
CALL CELLML_MODELS_FIELD_CREATE_FINISH(CELLML,ERR,ERROR,*999)
```

```
!Assign CellML model number to use at a host field dof in the CellML models field
CALL FIELD_PARAMETER_SET_UPDATE_GRID(CELLML_MODELS_FIELD,FIELD_VALUES_SET_TYPE, &
   & 1,1,1,FIELD_U_VARIABLE_TYPE,MODEL_USER_NUMBER,ERR,ERROR,*999)
!Assign model variable numbers to return to the host dof in the CellML models field
CALL CELLML_MODELS_VARIABLE_NUMBER_GET(CELLML,MODEL_USER_NUMBER, &
   MODEL_VARIABLE_URI,MODEL_VARIABLE_NUMBER,ERR,ERROR,*999)
CALL FIELD_PARAMETER_SET_UPDATE_GRID(CELLML_MODELS_FIELD,FIELD_VALUES_SET_TYPE, &
   & 1,1,2,FIELD_U_VARIABLE_TYPE,MODEL_VARIABLE_NUMBER,ERR,ERROR,*999)

!Setup the CellML state model field. Each state variable will be assigned to a
!field component in the CellML state models field.
CALL CELLML_MODEL_STATE_FIELD_CREATE_START(STATE_FIELD_USER_NUMBER,CELLML, &
   & CELLML_MODELS_STATE_FIELD,ERR,ERROR,*999)
!Finish creating the state variable field. N.b. Default field values are the
!default for the state variable.
CALL CELLML_MODEL_STATE_FIELD_CREATE_FINISH(CELLML,ERR,ERROR,*999)

!Change initial conditions etc. e.g., [Ca2+] to 1.0 mMol
CALL CELLML_MODELS_STATE_VARIABLE_COMPONENTS_GET(CELLML,MODEL_USER_NUMBER, &
   & Ca_URI,STATE_VARIABLE_COMPONENT_NUMBER,ERR,ERROR,*999)
CALL FIELD_PARAMETER_SET_UPDATE_GRID(CELLML_MODELS_STATE_FIELD, &
   & FIELD_VALUES_SET_TYPE,1,1,STATE_VARIABLE_COMPONENT_NUMBER, &
   & FIELD_U_VARIABLE_TYPE,1.0_DP,ERR,ERROR,*999)
```

```
!Setup varying parameters for each intermediate
CALL CELLML_MODEL_INTERMEDIATE_FIELD_CREATE_START(INTERMEDIATE_FIELD_USR_NUMBER,&
        & CELLML,CELLML_INTERMEDIATE_FIELD,ERR,ERROR,*999)
!Set the varying parameters
CALL CELLML_MODEL_INTERMEDIATE_ADD(CELLML,INTERMEDIATE_NUMBER/URI,IKr_URI, &
        & ERR,ERROR,*999)
!Finish the creation of the varying parameters.
CALL CELLML_MODEL_INTERMEDIATE_FIELD_CREATE_FINISH(CELLML,ERR,ERROR,*999)

!Setup varying parameters for each variant
CALL CELLML_MODELS_PARAMETERS_CREATE_START(CELLML,ERR,ERROR,*999)
!Set the varying parameters
CALL CELLML_MODELS_PARAMETERS_VARYING_ADD(CELLML,MODEL_USER_NUMBER,gKr_URI, &
        & ERR,ERROR,*999)
!Finish the creation of the varying parameters.
CALL CELLML_MODELS_PARAMETER_CREATE_FINISH(CELLML,ERR,ERROR,*999)


!Setup the CellML variants parameter values field. The number of components in
!this field will be equal to the sum of the number of varying parameters across !all
models.
CALL CELLML_MODELS_PARAMETER_FIELD_CREATE_START(PARAM_FIELD_USER_NUMBER,CELLML, &
        & CELLML_PARAM_FIELD,ERR,ERROR,*999)
!Get gKr parameter component
CALL CELLML_MODELS_PARAMETER_FIELD_COMPONENT_GET(CELLML,MODELS_USER_NUMBER/URI, &
        & gKr_USER_NUMBER/gKr_URI,gKr_COMPONENT_NUMBER,ERR,ERROR,*999)
!Set field properties
CALL FIELD_COMPONENT_INTERPOLATION_SET(CELLML_PARAM_FIELD,gKr_COMPONENT_NUMBER, &
        & NODE_BASED_INTERPOLATION,ERR,ERROR,*999)
```

```
!Finish creating the models parameter field. This will then create an embedded
!field which contains the working data at the level of the host field dofs. It !will
contain, for example, the values of the parameters interpolated at the host
!dofs and any extra components for integrator working data.
CALL CELLML_MODELS_PARAMETER_FIELD_CREATE_FINISH(CELLML,ERR,ERROR,*999)
!Change default parameters etc.
CALL FIELD_PARAMETER_SET_UPDATE_NODE(PARAM_FIELD,FIELD_VALUES_SET_TYPE,1,1, &
        & gKr_COMPONENT_NUMBER,FIELD_STANDARD_VARIABLE_TYPE,0.05_DP, &
        & ERR,ERROR,*999)


!Generate/instatiate the CellML i.e., compile, link etc. Would first check for !valid
setup.
CALL CELLML_GENERATE(CELLML,ERR,ERROR,*999)
```

Bidomain equations

$$V_m = \varphi_i - \varphi_e$$

$$A_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot \left(\sigma_i \nabla V_m \right) = \nabla \cdot \left(\sigma_i \nabla \varphi_e \right) - A_m \left(I_{ion} - I_m \right) + i_i$$

W

$$\nabla \cdot \left(\left(\sigma_e + \sigma_i \right) \nabla \varphi_e \right) = - \nabla \cdot \left(\sigma_i \nabla V_m \right) - i_i - i_e$$

- $_i$ = intracellular potential
- $_e$ = extracellular potential
- A_m = membrane surface/volume ratio
- C_m = membrane capacitance
- $_e$ = intracellular conductivity tensor
- ▪ $_i$ = intracellular conductivity tensor
- I_{ion} = ionic source current from cellular model
- I_m = transmembrane stimulus current
- i_i = current injection per volume intro intracellular space
- i_e = current injection per volume intro extracellular space