DAE Tools Software
Overview

D.D. Nikolić

Updated: 26 March 2019

DAE Tools Project, http://www.daetools.com

What is DAE Tools?

Equation-based Object-oriented modelling, simulation, and optimisation software.

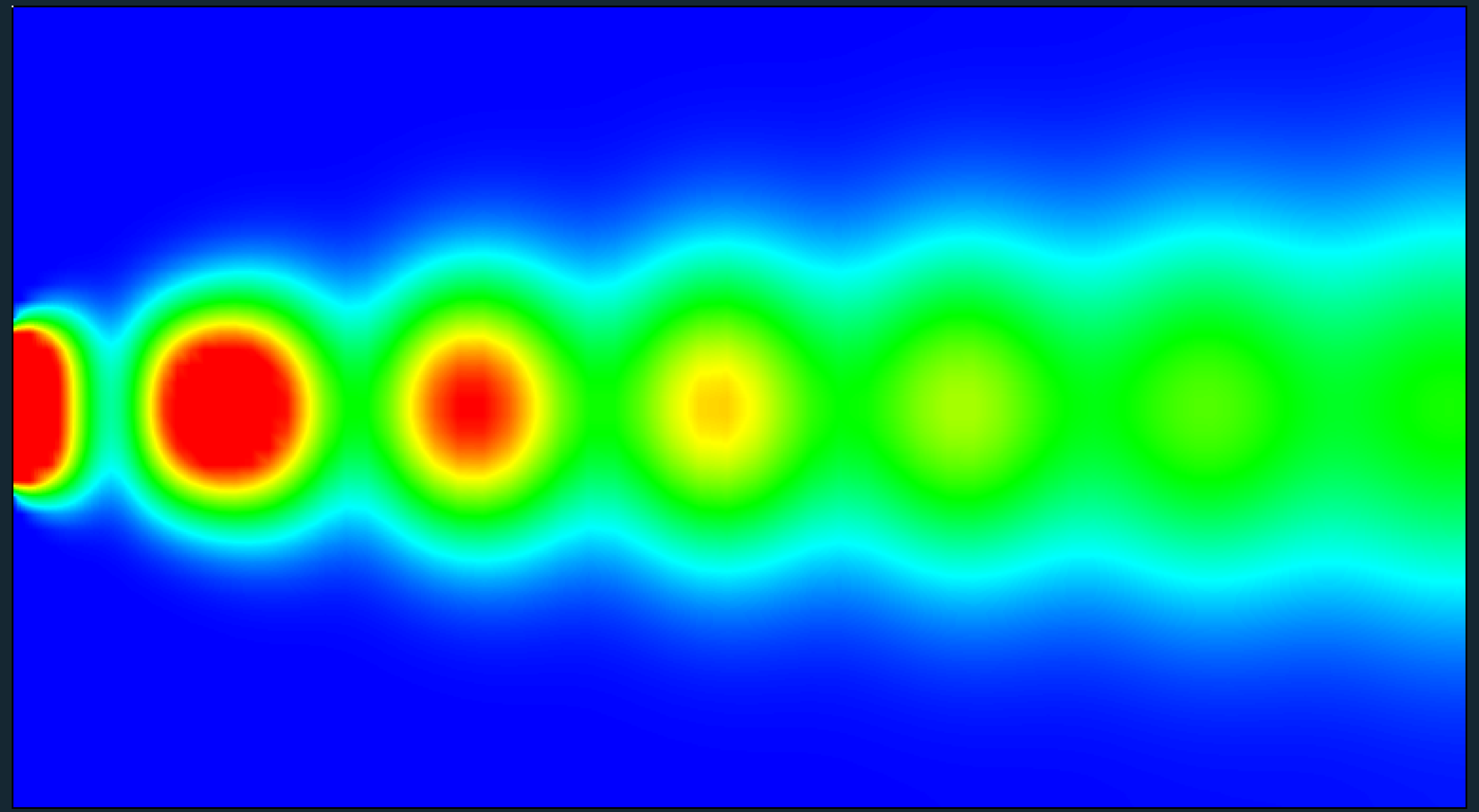
Areas of application:

- Initially: chemical process industry (mass, heat and momentum transfers, chemical reactions, separation processes, thermodynamics, electro-chemistry)
- Nowadays: **multi-domain**

Free/Open source software (GNU GPL).

Cross-platform (GNU/Linux, Windows, MacOS).

Multiple architectures (32/64 bit x86, ARM, ...).



Convective heat-transfer

What is DAE Tools?

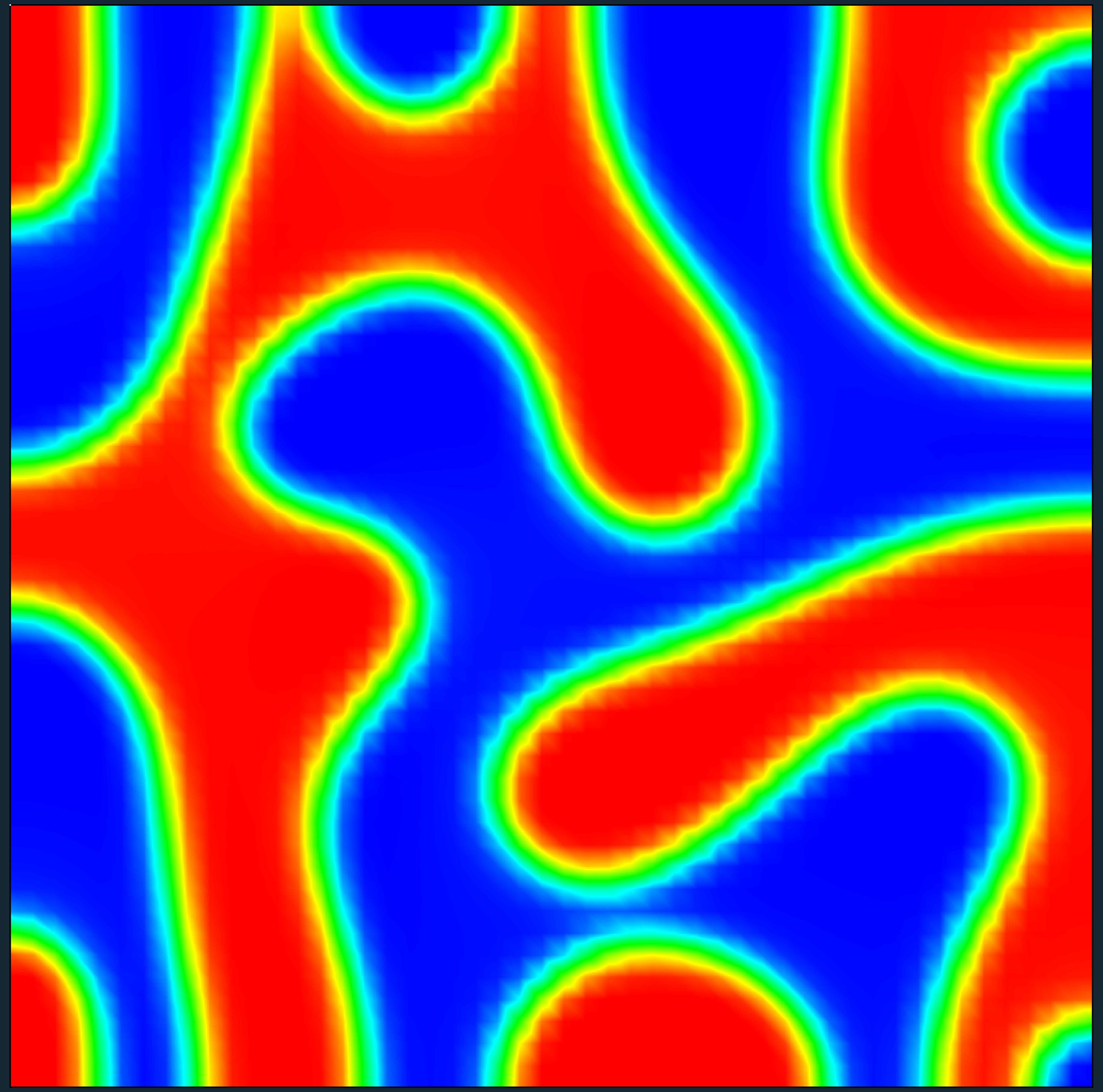
DAE Tools is not:

A modelling language nor a collection of numerical libraries.

DAE Tools is:

A higher level structure – an architectural design of interdependent software components providing an API for:

- Model development/specification
- Activities on developed models: simulation, sensitivity analysis, optimisation, and parameter estimation
- Processing of the results
- Report generation
- Code generation, co-simulation & model exchange



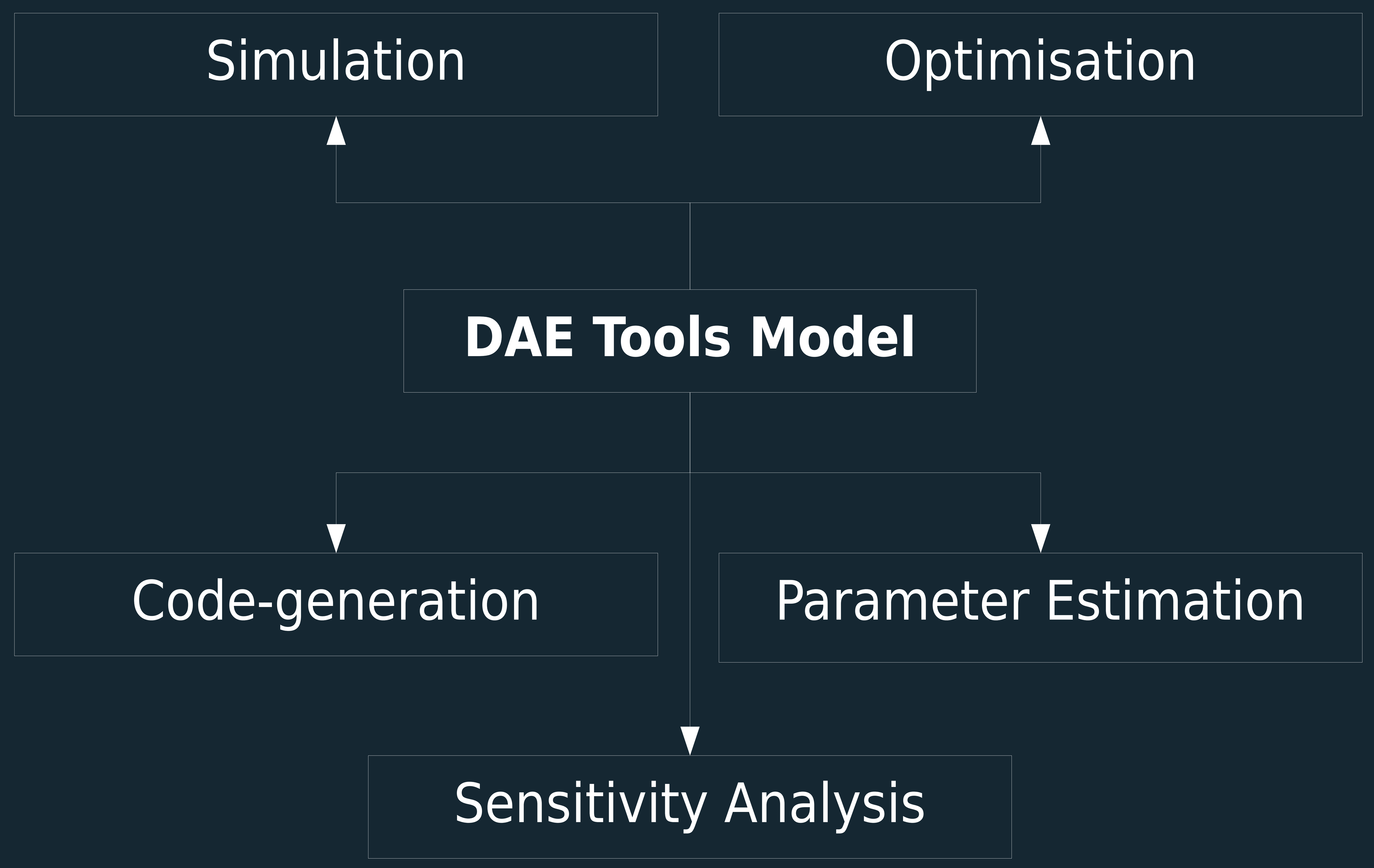
Cahn-Hilliard equation

What can be done with DAE Tools?

Modelling of complex multiscale/multiphysics processes/phenomena with complex schedules.

Single model definition as a basis for all activities:

- **Simulation** (steady-state & transient)
- **Optimisation** (NLP/MINLP)
- **Sensitivity Analysis** (local and global)
- **Parameter Estimation**
- **Code-generation** & **co-simulation**



Types of systems that can be modelled

Initial value problems of implicit form:

- Described by a system of linear, non-linear and partial-differential equations
- Continuous with some elements of event-driven systems (i.e. discontinuous equations, state transition networks, discrete events)
- Steady-state or dynamic
- With lumped or distributed parameters (FD, FV, FE)
- Index-1 DAE systems only

Steady-State
Dynamic

Continuous
Event-Driven

With Lumped Parameters
With Distributed Parameters

The Hybrid Approach

DAE Tools apply a **hybrid approach** between **modelling** and **general purpose** programming languages.

The hybrid approaches **combines** the **strengths** of **both approaches**:

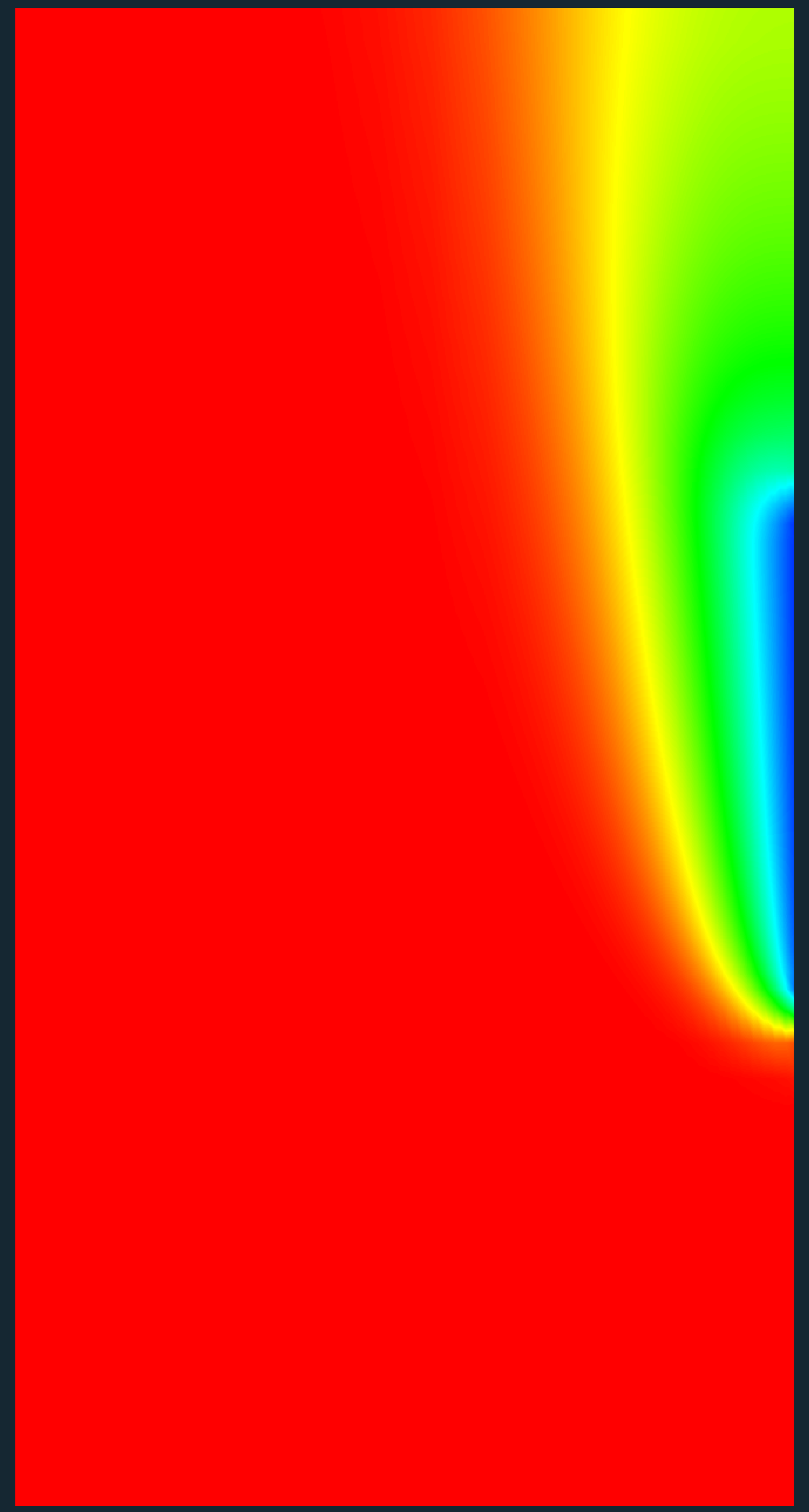
- Developed in C++ for performance
- Key modelling concepts provided by the API
- Python wrappers for model development, execution of simulations and all other tasks

```
class BufferTank(daeModel):
  def __init__(self, Name, Parent = None, Description = ""):
    daeModel.__init__(self, Name, Parent, Description)

    self.Density = daeParameter("Density", kg/m**3, self)
    self.Area    = daeParameter("Area",    m**2,    self)
    self.Alpha   = daeParameter("Alpha",   unit(),  self)

    self.HoldUp  = daeVariable("HoldUp",  mass_t,     self)
    self.FlowIn  = daeVariable("FlowIn",  flowrate_t, self)
    self.FlowOut = daeVariable("FlowOut", flowrate_t, self)
    self.Height  = daeVariable("Height",  length_t,   self)

  def DeclareEquations(self):
    # Mass balance
    eq = self.CreateEquation("MassBalance")
    eq.Residual = self.HoldUp.dt() - self.FlowIn() + self.FlowOut()

    # Relation between liquid level and holdup
    eq = self.CreateEquation("LiquidLevelHoldup")
    eq.Residual = self.HoldUp() - self.Area()*self.Height()*self.Density()

    # Outlet flowrate as a function of the liquid level
    eq = self.CreateEquation("OutletFlowrate")
    eq.Residual = self.FlowOut() - self.Alpha() * Sqrt(self.Height())
```

Why YET ANOTHER modelling software?

The combination of the features of modelling and general-purpose programming languages in the **Hybrid approach** provide the following capabilities:

- **Runtime model generation**
- **Runtime simulation set-up**
- **Complex schedules**
- **Interoperability** with the **third-party software**
- Suitability for **embedding** and use as **a web application** or **software as a service**
- **Code-generation**, model exchange and co-simulation



Parallel-plate reactor with an active surface

Programming paradigms

Equation-based (acausal) approach

- Equations given in an implicit form (as a residual)
- Input-output causality is not fixed
    - Increased model re-use
    - Different simulation scenarios based on a single model by specifying different degrees of freedom

Object-oriented approach

- Everything is an object (variables, equations, models ...)
- All objects can be manipulated in runtime
- All C++/Python object-oriented concepts supported
- The hierarchical model decomposition

Single definition (acausal equation):

$$x_1 + x_2 + x_3 = 0$$

But, three simulation scenarios:

$$a)\ x_1 = -x_2 - x_3;\ \ for\ fixed\ x_2\ and\ x_3$$

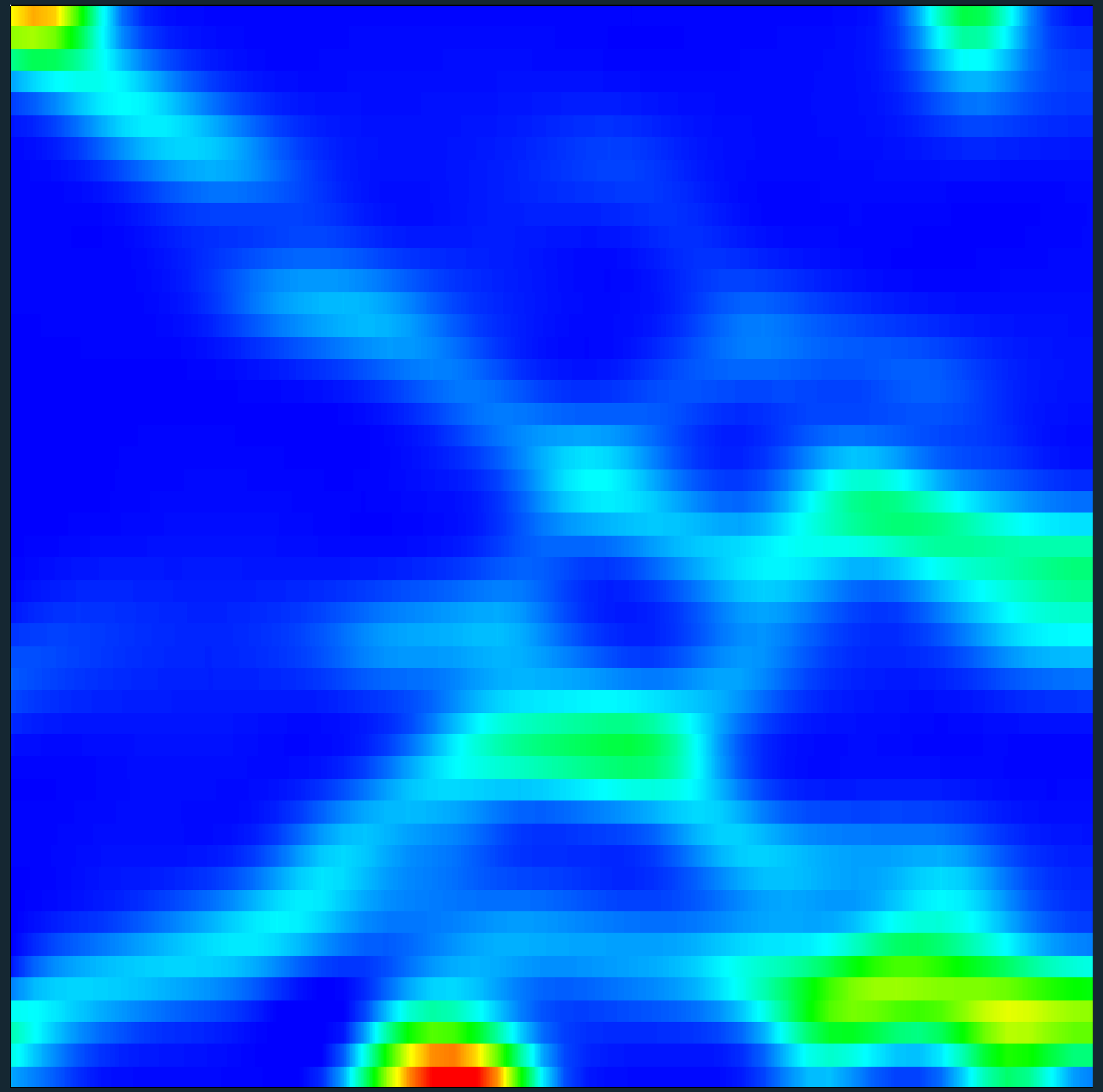
$$b)\ x_2 = -x_1 - x_3;\ \ for\ fixed\ x_1\ and\ x_3$$

$$c)\ x_3 = -x_1 - x_2;\ \ for\ fixed\ x_1\ and\ x_2$$

Multiphysics capabilities

Model multiple simultaneous physical phenomena using the finite difference, finite volume and finite element methods

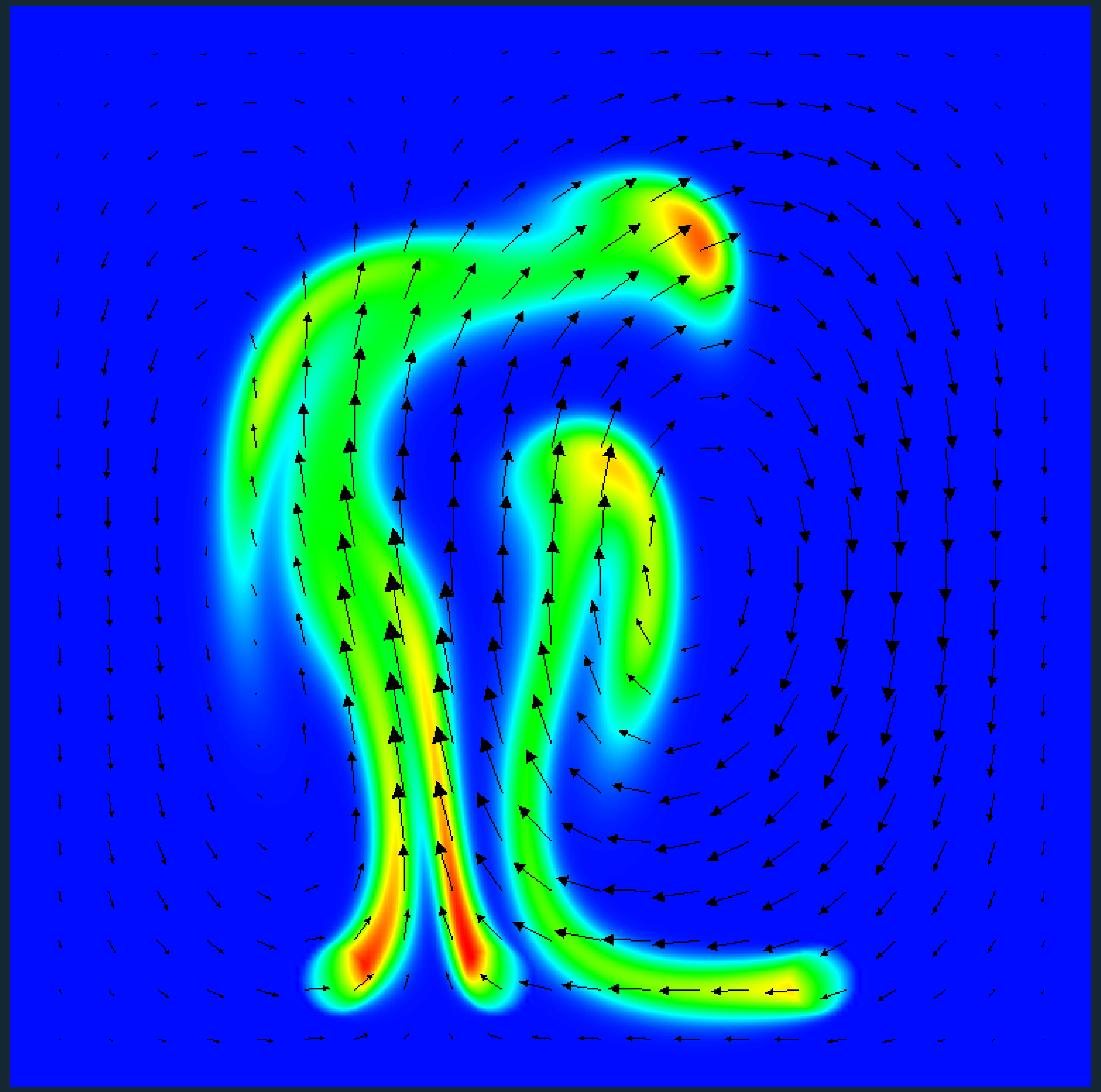
- DAE Tools utilise deal.II library to generate a set of differential equations for given inputs (mesh, FE space, weak form, BCs, ...)
- Unique features:
  - Generate several non-linear FE systems in the same model
  - Mix with the other equations in the model (i.e. FV)
  - Use DAE Tools variables to set boundary conditions, evaluate source terms and non-linear coefficients
  - Impose constraints and add any number of auxiliary equations
- Explore tutorials models (Cahn-Hilliard equation, convective heat tranfer, flow in porous media, ...)



Flow in porous media

Parallel computation

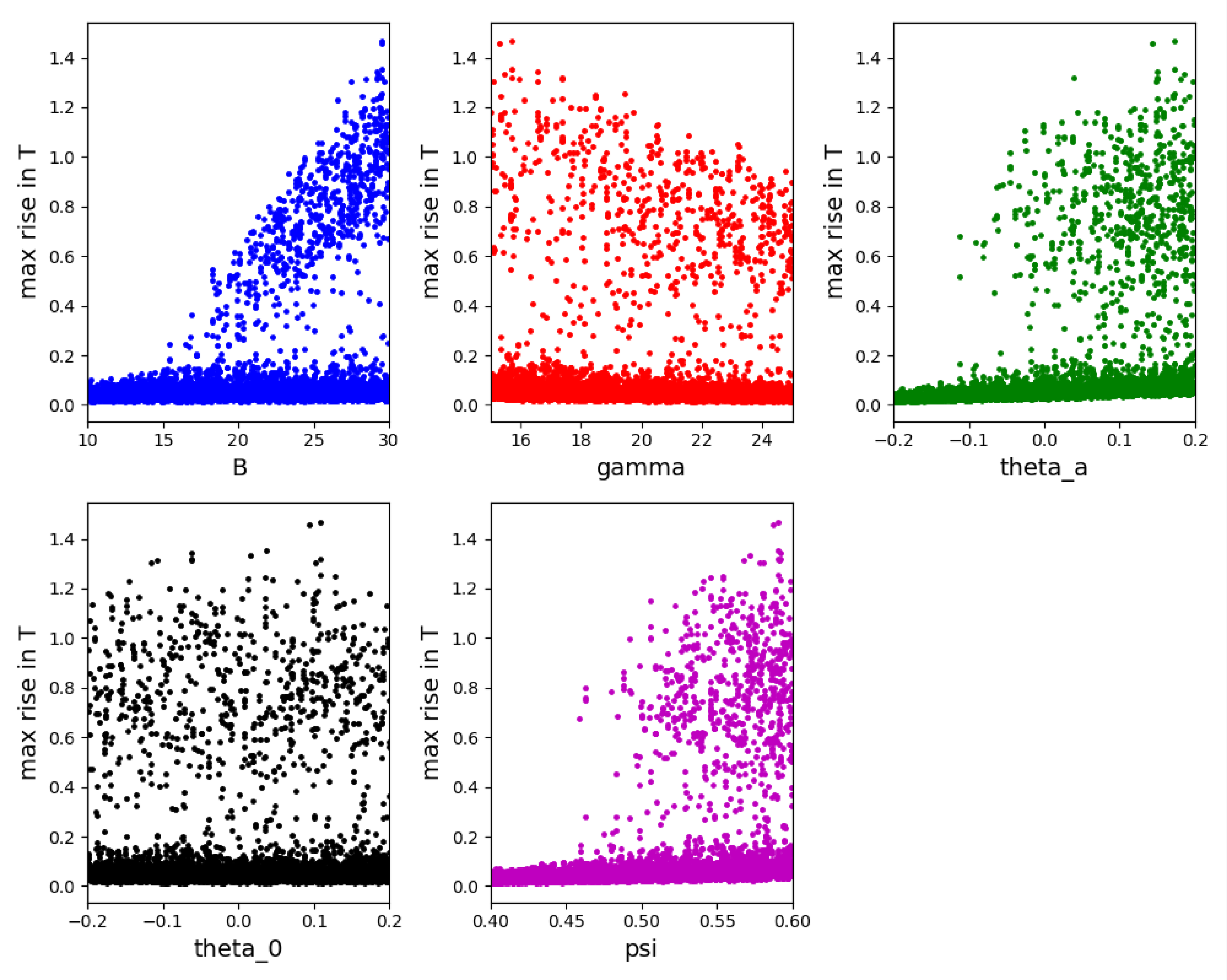
- The shared-memory parallel programming model
- Export to OpenCS models for simulation on distributed memory systems
- OpenCS utlised for parallel evaluation of model equations
  - OpenMP: general purpose processors and manycore devices
  - OpenCL: streaming processors (GPU, FPGA) and heterogeneous systems (CPU+GPU, CPU+FPGA)
- Assembly of Finite Element systems (OpenMP)
- Solution of systems of linear equations (SuperLU_MT, Pardiso and Intel Pardiso solvers)
- Global Sensitivity Analysis (multiprocessing.Pool)



Transient Stokes flow driven by the differences in buoyancy

Sensitivity analysis

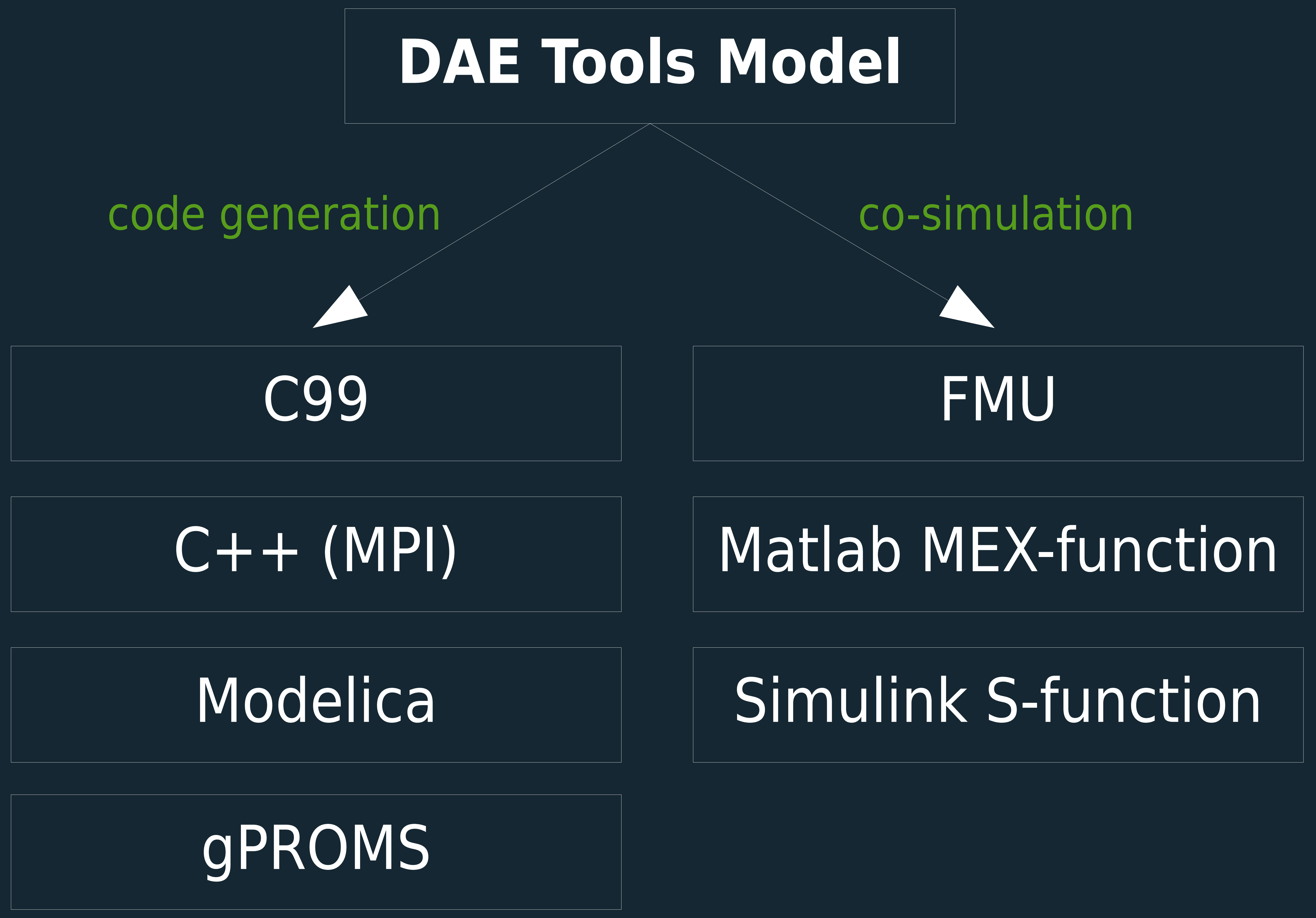
- **Local** sensitivity analysis (derivative-based)
- **Global** sensitivity analysis (SALib library):
  - **1st and 2nd order sensitivities** and confidence intervals
  - **Total sensitivity indices** and confidence intervals
  - **Scatter plots**
- Methods available:
  - **Method of Morris** (elementary effect method)
  - **FAST** (variance-based)
  - **Sobol** (variance-based)
- Simulations performed in parallel (multiprocessing.Pool)



SA scatter plot

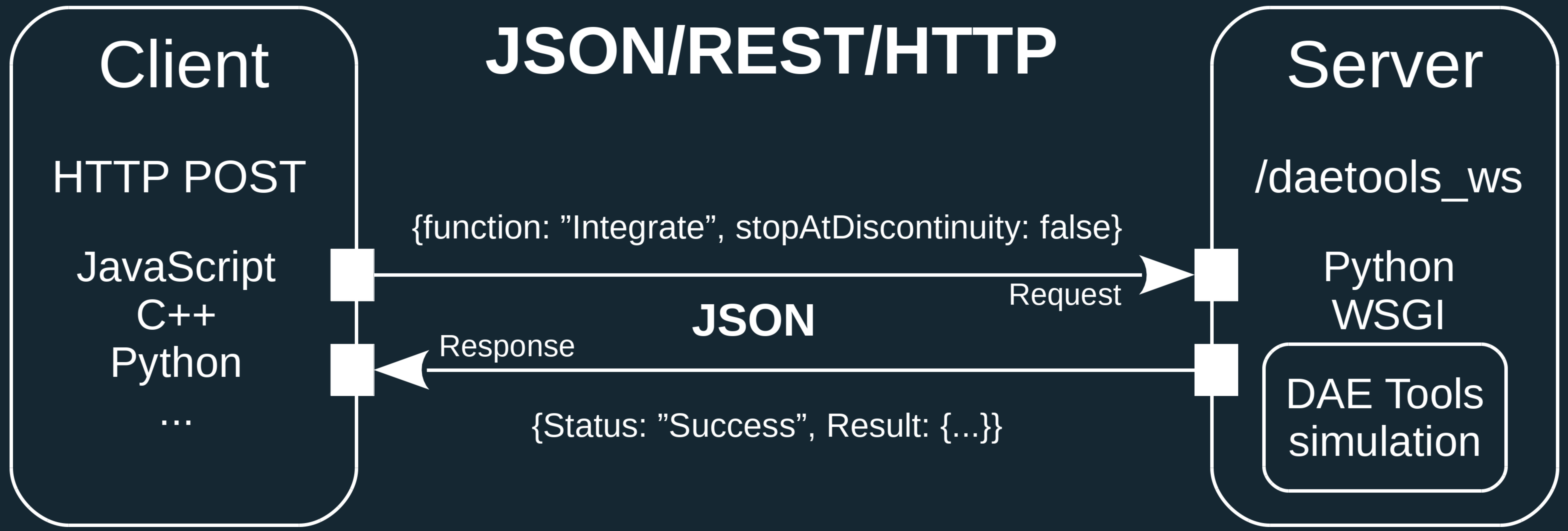
Code generation & co-simulation

- **Code-generation**
  - Modelica
  - gPROMS
  - C99 (embedded systems)
  - C++ MPI (distributed systems)
- **Co-simulation**
  - Matlab MEX-functions
  - Simulink user-defined S-functions
  - Functional Mockup Interface (FMI) for Co-Simulation



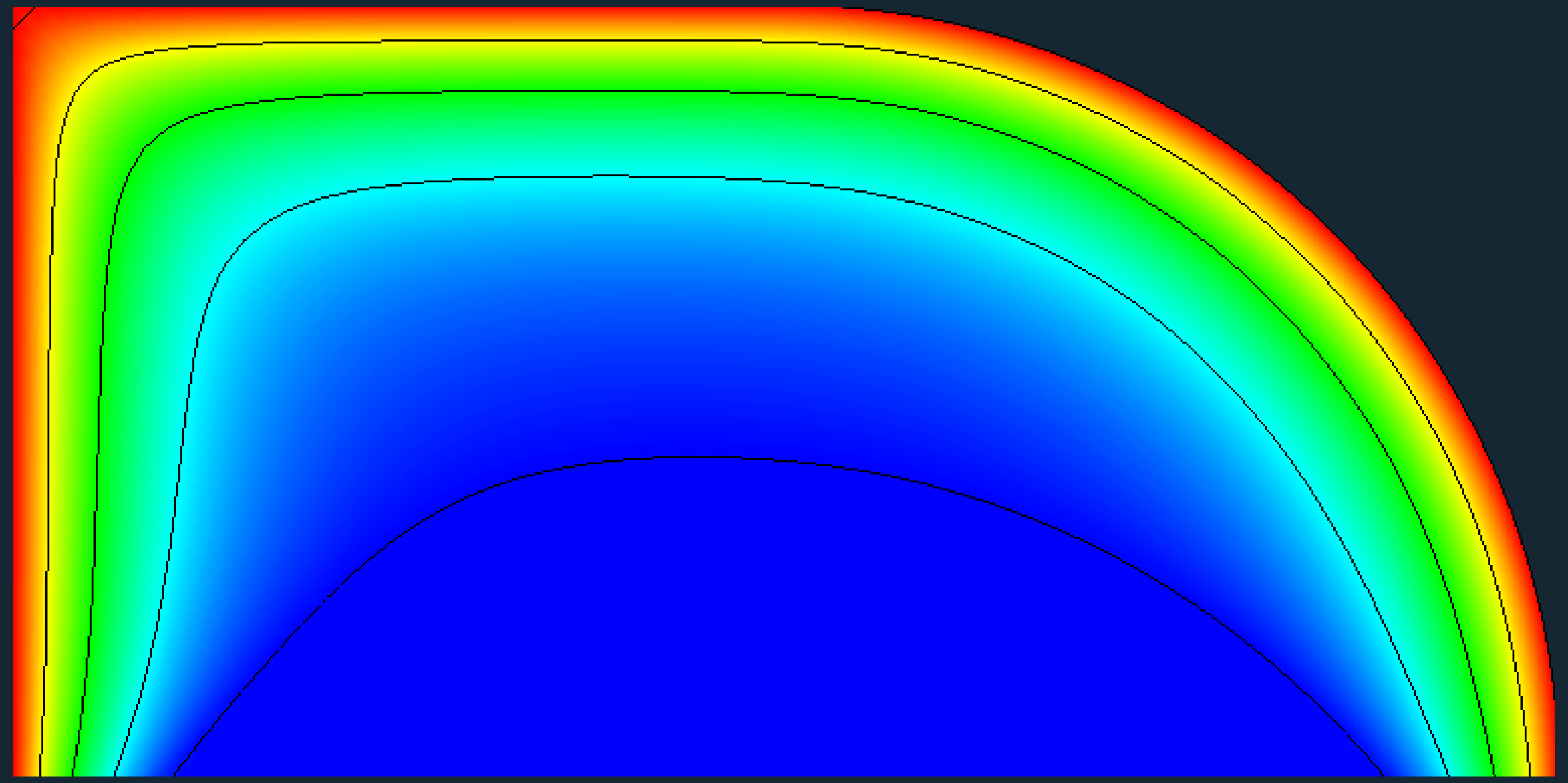
Software as a service

- **Web service with the RESTful API**
  - DAE Tools simulations (daetools_ws)
  - DAE Tools FMU objects (daetools_fmi_ws)
- **Language independent** (JavaScript, Python, C++, ...)
- **Benefits**:
  - Application servers
  - Individual simulations as a web service
  - Attractive Graphical User Interface



Additional features

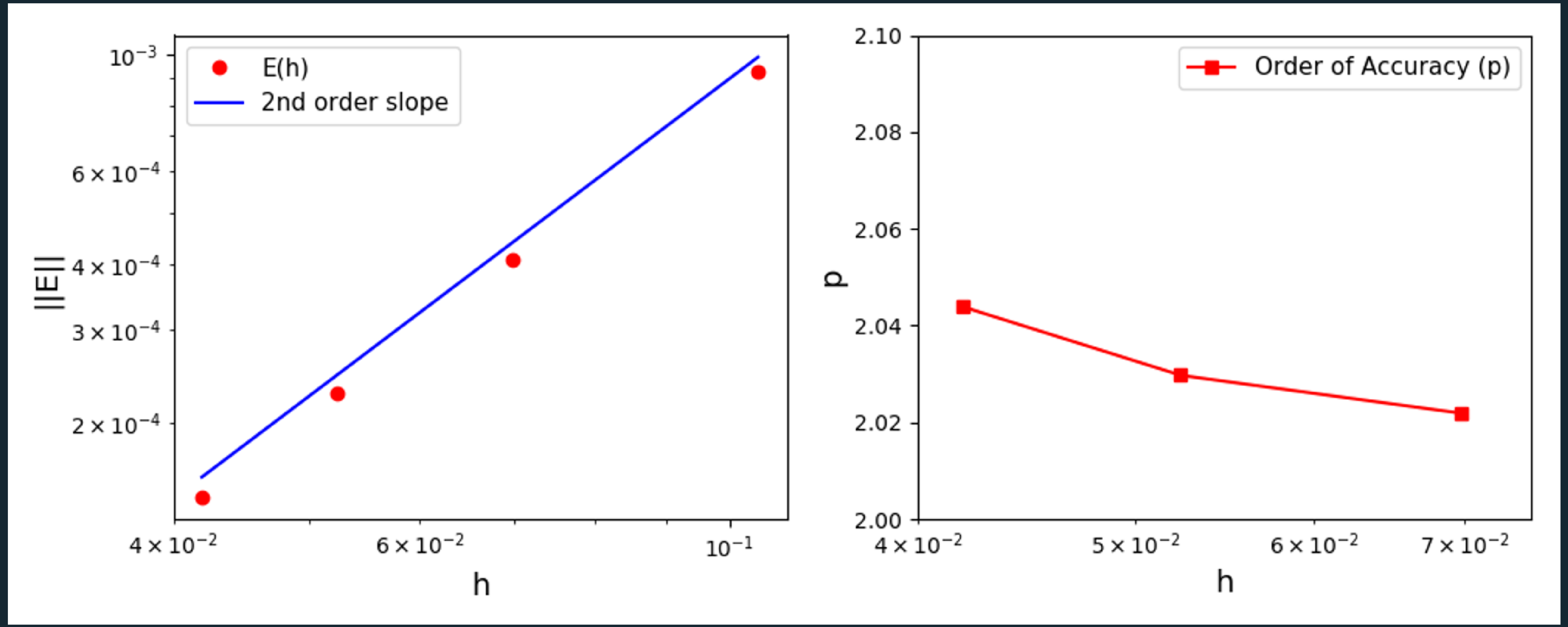
- Automatic differentiation (ADOL-C)
- Large number of the state-of-the-art solvers:
  - DAE (Sundials IDAS)
  - LA (SuperLU, SuperLU_MT, Trilinos Amesos/AztecOO, Pardiso, Intel Pardiso)
  - (MI)NLP (Ipopt, Bonmin, NLopt)
- Generation of model reports (XML + MathML, Latex)
- Export of simulation results to several file formats (csv, Matlab, Excel, json, xml, HDF5, Pandas, VTK)



Diffusion and reaction in a catalyst flake

Code verification

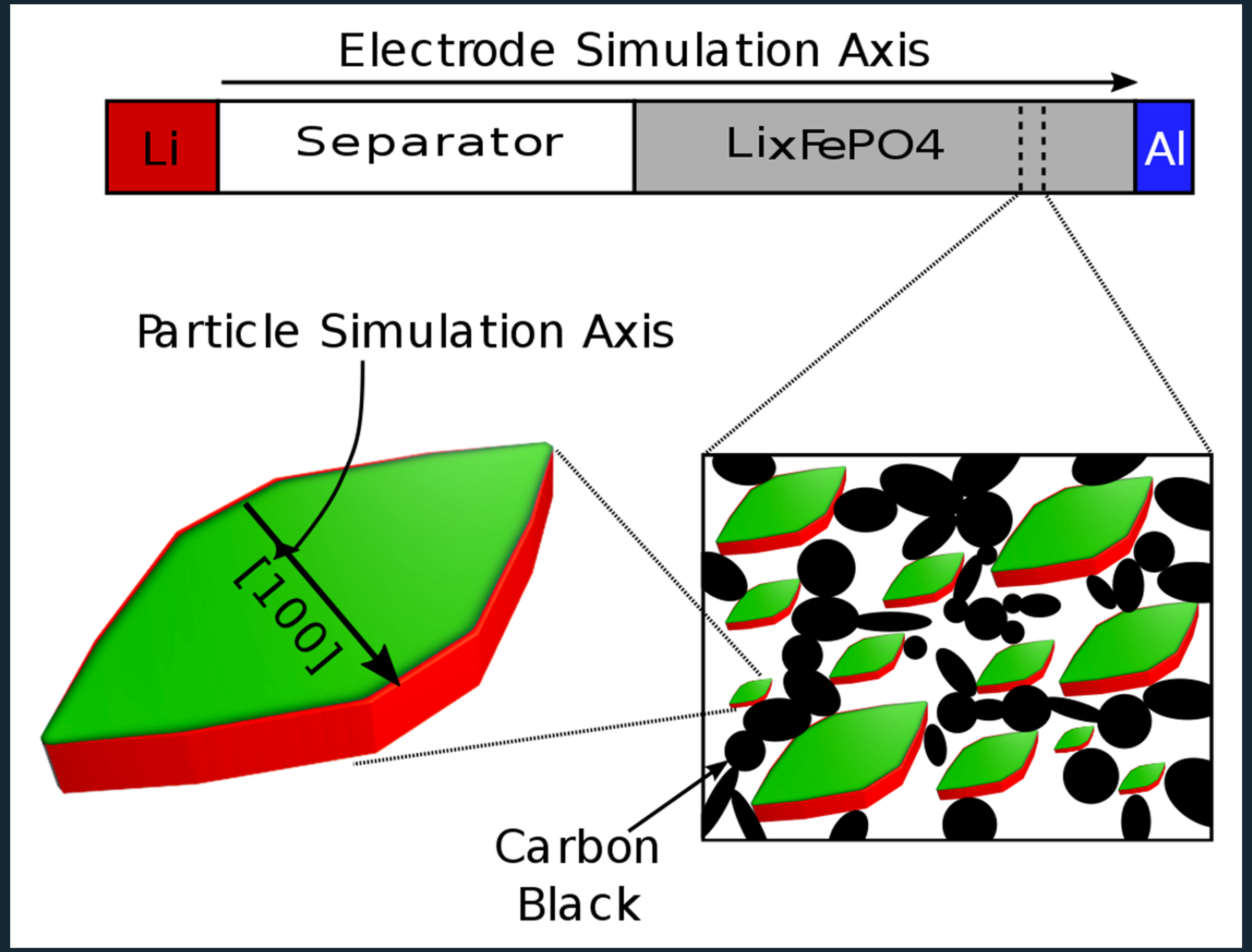
- **The formal code verification techniques** applied to test almost all aspects of the software
- The code verification methods used:
  - The **Method of Exact Solutions** (MES)
  - The **Method of Manufactured Solutions** (MMS)
- **The most rigorous acceptance criteria** used:
  - Percent Error
  - Consistency
  - Order-of-accuracy



Normalised global error and order-of-accuracy

Applications & case studies

- **Chemical engineering**: chemical reactions, separations...
- **Finite Elements**: heat transfer, Cahn-Hiliard equation, ...
- **Multi-scale problems**: multiphase porous electrodes, phase separating hydroxide-exchange fuel cells, PSA
- **Sensitivity analysis**: thermal analysis of a batch reactor and exothermic reaction
- **Optimisation**: Large-scale Constrained Optimisation Problem Set (COPS)
- **Domain Specific Languages**, **Embedded simulators** and **Web services**: DAE Tools (daetools_ws), NineML



Multi-scale model of phase-separating battery electrodes