Implementational Challenges in Localized Reduced Basis Multiscale Methods

Sven Kaulmann, PDESoft 2012, Münster

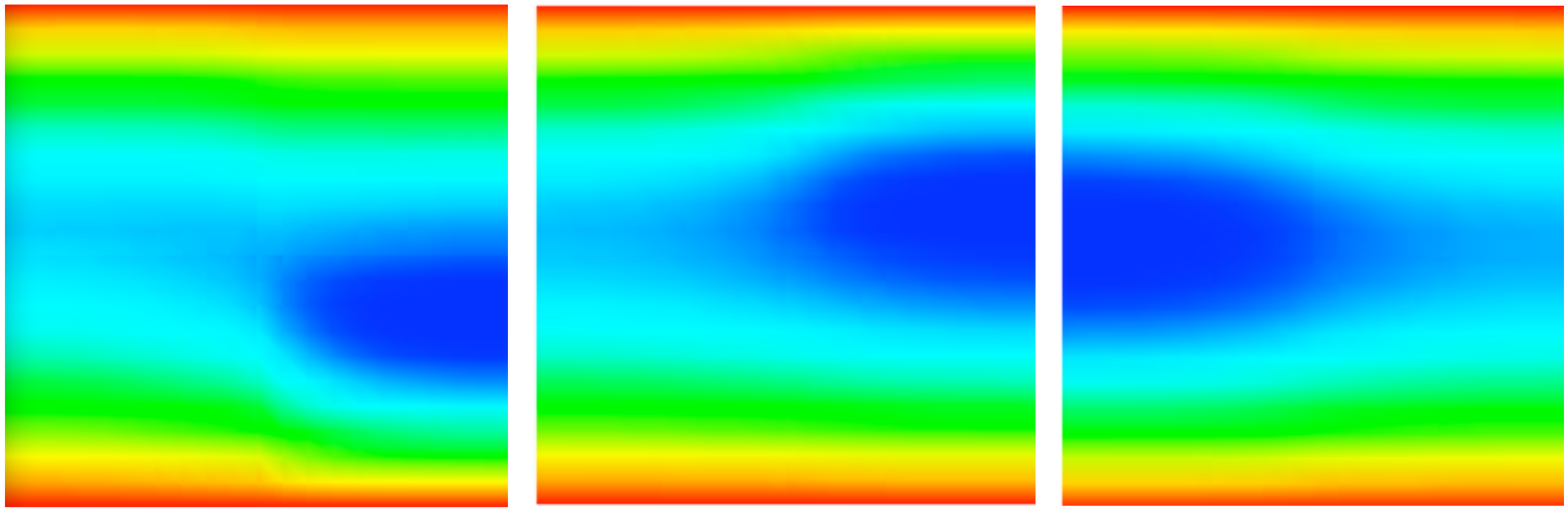
Joint work with
Felix Albrecht, Martin Drohmann,
Bernard Haasdonk, Mario Ohlberger

Sketch

$$\boldsymbol{B}(p_h, v_h, \mu) = \boldsymbol{L}(v_h, \mu) \quad \forall v_h \in S_h.$$

Sketch

$$\boldsymbol{B}(p_h, v_h, \mu) = \boldsymbol{L}(v_h, \mu) \quad \forall v_h \in S_h.$$

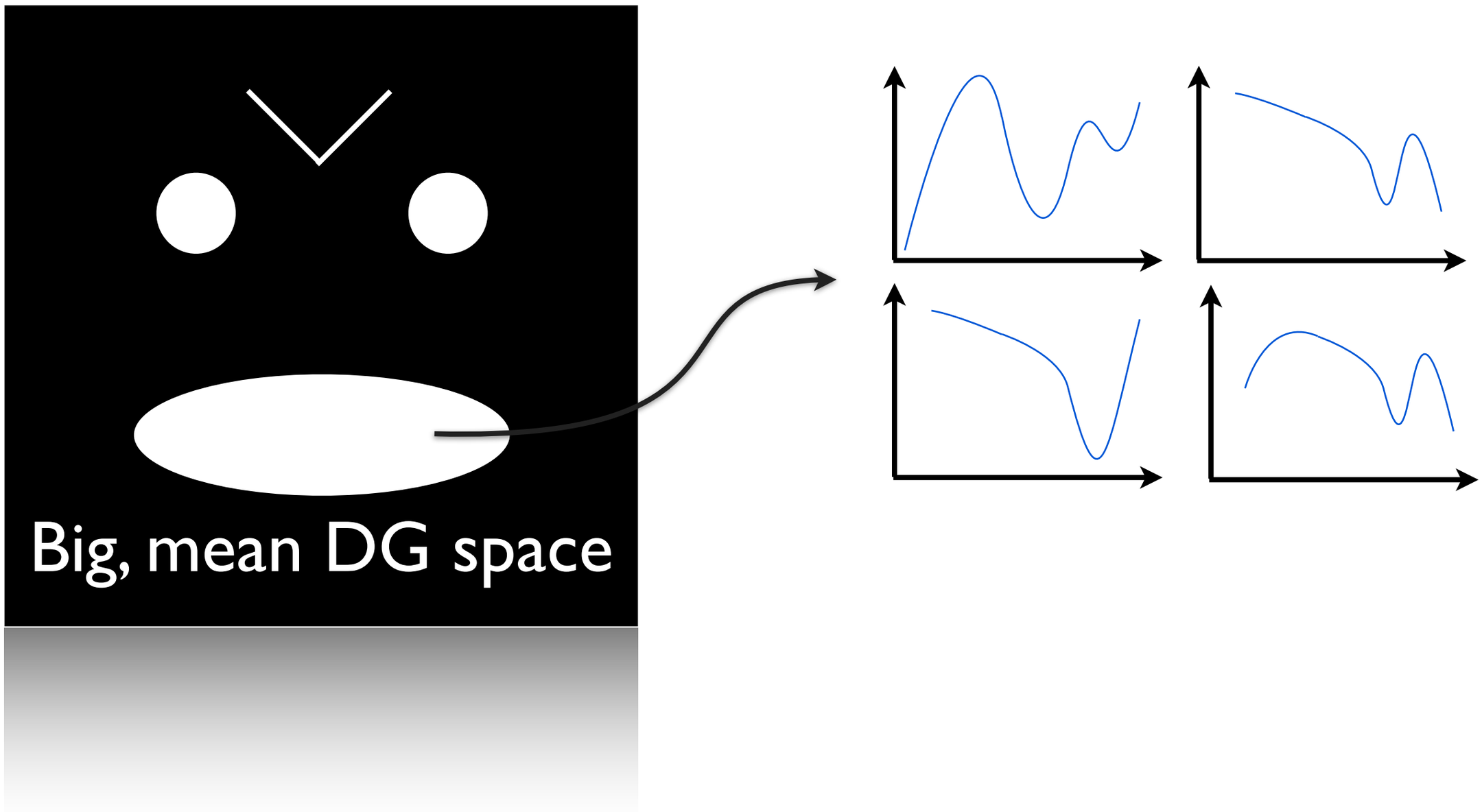


Sketch

$$\boldsymbol{B}(p_h, v_h, \mu) = \boldsymbol{L}(v_h, \mu) \quad \forall v_h \in S_h.$$

Sketch

$$\boldsymbol{B}(p_h, v_h, \mu) = \boldsymbol{L}(v_h, \mu) \quad \forall v_h \in S_h.$$



Sketch

$$\boldsymbol{B}(p_h, v_h, \mu) = \boldsymbol{L}(v_h, \mu) \quad \forall v_h \in S_h.$$

Big, mean DG space

Small, nice RB space

Sketch

$$\boldsymbol{B}(p_h, v_h, \mu) = \boldsymbol{L}(v_h, \mu) \quad \forall v_h \in S_h.$$

$$\boldsymbol{B}(p_N, v_N, \mu) = \boldsymbol{L}(v_N, \mu) \quad \forall v_N \in S_N.$$

Introduction

- Reduced Basis Methods: Model reduction for parametrized partial differential equations
- Used in many-query or real-time contexts
- Construct a reduced surrogate of a high-dimensional discretization space (FE, DG) by solving the full equation for some parameter samples
- Assumption: All data functions depend affinely on the parameter

$$f(x, \boldsymbol{\mu}) = \sum \Theta_i(\boldsymbol{\mu}) f_i(x)$$

Introduction

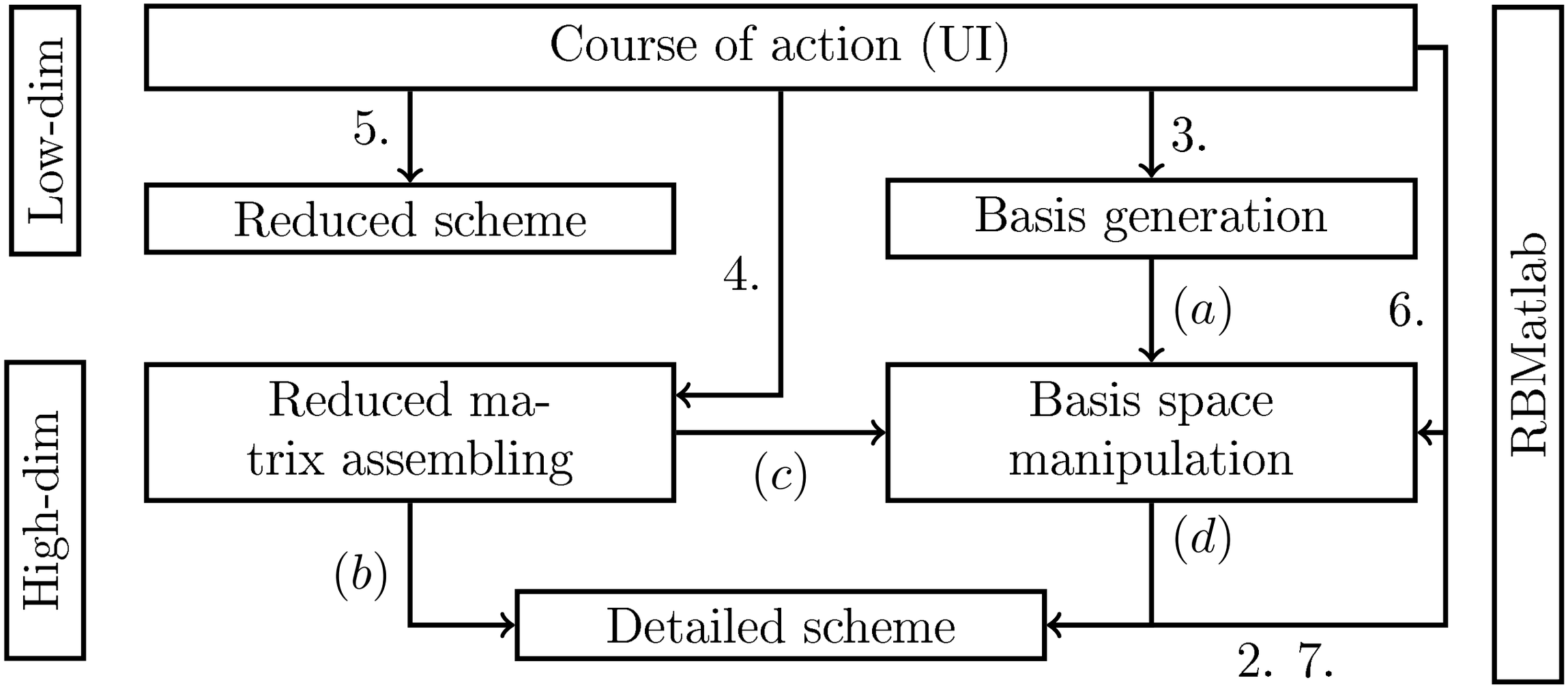
- Most important: Offline-Online decomposition:

Offline	Online
• Computation of reduced basis, estimator • Projection • h-dependent	• All computations in reduced space • h-independent • very rapid

dune-rb

- Based on DUNE
- Clean and preferably small interfaces that enable users to apply RB to their own discretizations
- Modularity: users only need to implement very few interfaces for their problem
- Implementation of Reduced Basis methods based only on Linear Algebra quantities (matrices, vectors)
- Strict separation between offline and online phase
- Links to Matlab and Python
- Contributors: F. Albrecht, M. Drohmann, S. Kaulmann

dune-rb Overview



Linear Algebra

Universität Stuttgart

SimTech
Cluster of Excellence

Separable Parametric Container

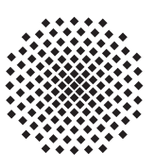
- Provide an easy way of representing separable parameter-dependent quantities such as

$$\underline{A}((\mu_1, \mu_2, \mu_3)^\tau) = \mu_1 \mu_2 \underline{A_1} + \mu_3^2 \underline{A_2}$$

- Interoperability with Matlab (symbolic coefficients)
- Current Implementation:
  - Coefficients using math expression parser by Yann Ollivier
  - Matrices using Eigen (eigen.tuxfamily.org)

Separable Parametric Container

```
namespace Dune {
namespace RB {
namespace LA {
namespace SeparableParametric {
namespace Container {
class Interface
{
public:
  Interface(const int rows, const int cols);

  template< class ParamType >
  double coefficient(const int q, const ParamType param) const;

  template< class ParamType >
  CoefficientsVectorType coefficients(const ParamType param) const;

  ComponentType& component(const int q);

  int numComponents() const;

  template< class ParameterType >
  const CompleteType& complete(const ParameterType param) const;

  const std::vector< std::string >& getSymbolicCoefficients() const;

  void setSymbolicCoefficients(std::vector< std::string > symbolicCoefficients,
                               const std::string variable = "mu");

  inline bool save(const std::string& path) const;
  inline bool load(const std::string& path);
}
```



Separable Parametric Container

```
namespace Dune {
namespace RB {
namespace LA {
namespace SeparableParametric {
namespace Container {
class Interface
{
public:
  Interface(const int rows, const int cols);

  template< class ParamType >
  double coefficient(const int q, const ParamType param) const;

  template< class ParamType >
  CoefficientsVectorType coefficients(const ParamType param) const;

  ComponentType& component(const int q);

  int numComponents() const;

  template< class ParameterType >
  const CompleteType& complete(const ParameterType param) const;

  const std::vector< std::string >& getSymbolicCoefficients() const;

  void setSymbolicCoefficients(std::vector< std::string > symbolicCoefficients,
                               const std::string variable = "mu");

  inline bool save(const std::string& path) const;
  inline bool load(const std::string& path);
}
```

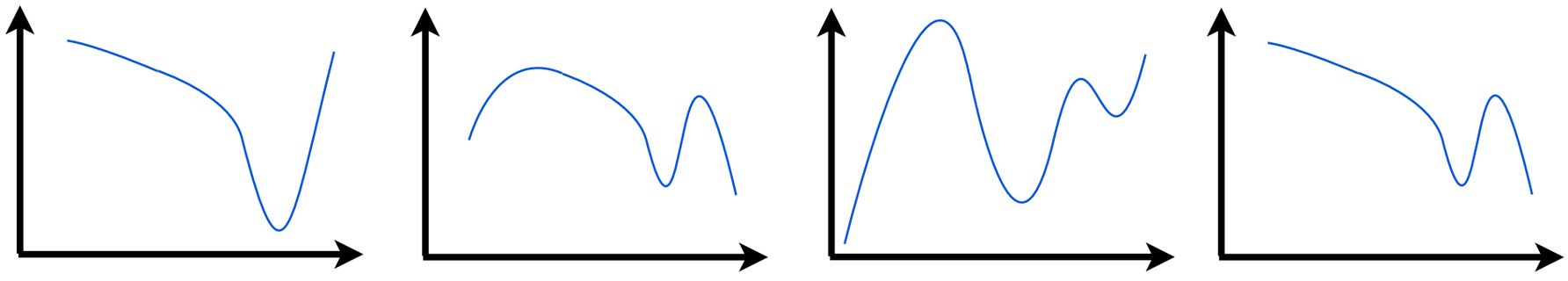
Example:

```
setSymbolicCoefficients(("mu[0]*mu[1]", "mu[2]*mu[2]"))
```

Offline

Reduced Basis Generator

```
namespace Dune { namespace RB { namespace Offline {
namespace Generator { namespace ReducedBasis {
template< class OfflineSpaceImp >
class Interface
{
  Interface(OfflineSpaceType& space);

  void init(const ParameterType& param);

  template< class ParameterSampleType >
  const boost::uuids::uuid generate(const ParameterSampleType& paramSample);
}; }}}}}
```



Reduced Basis Space

- No function space in the sense of Dune::FEM or PDELab function spaces, only dealing with vectors, matrices

Reduced Basis Space

- No function space in the sense of Dune::FEM or PDELab function spaces, only dealing with vectors, matrices

```
namespace Dune { namespace RB { namespace Offline { namespace Space {
template< class CoarseMapperImp >
class Interface
{
  int size() const;

  const DofMatrixType& getDofMatrix() const;
  const DofVectorType getDofVector(const int i) const;

  void setDofMatrix(const DofMatrixType& dofMatrix);
  void setDofVector(const int i, const DofVectorType& dofVector);

  void addDofMatrix(const DofMatrixType& dofMatrix);
  void addDofVector(const DofVectorType& dofVector);

  inline const MapperType& mapper() const;

  inline bool save(const std::string& path) const;

  inline bool load(const std::string& path);

}; }}}}
```

Generator Concept

- Perform high-dimensional computation (e.g. projection of system matrices)
- Generate low-dimensional quantities
- Example: Reduced solver generator, estimator generator

```
namespace Dune { namespace RB { namespace Offline {
namespace Generator { namespace ReducedData { namespace Solver {
class Interface
{
  const ReducedSolverConnectorType& generate();
  void update();

}; }}}}}}
```

Online

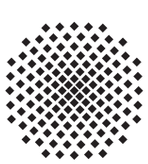
Connector-Concept

- Contains only reduced quantities
- Can be saved to and loaded from disk
- Assembled by generators

Connector-Concept

- Contains only reduced quantities
- Can be saved to and loaded from disk
- Assembled by generators

```
namespace Dune { namespace RB { namespace Reduced { namespace Solver { namespace Connector {
class Interface
{

  DofVectorType solve(const ParameterType& param);


  bool save(const std::string& path) const;
  bool load(const std::string& path);

}; }}}}}}
```



Detailed

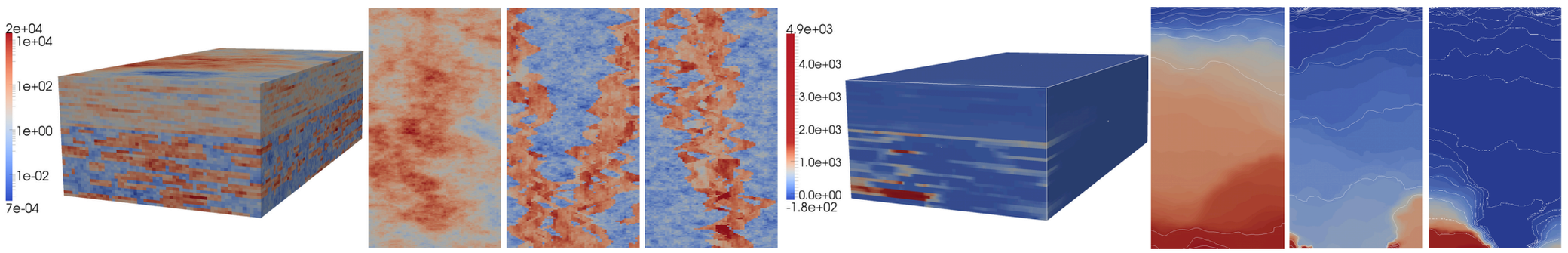
Detailed Solver Connector

- Interface to your high-dimensional solver implementation, non-intrusive

```
namespace Dune { namespace RB { namespace Detailed {
namespace Solver { namespace Connector {
class Interface
{
  const ModelType& model() const;

  void init();

  const MatrixType& getSystemMatrix() const;

  const VectorType& getRightHandSide() const;

  void visualize(const DofVectorType& vector) const;

  bool solve(const ParameterType& param,
             DofVectorType& solution);

  const MapperType& mapper() const;

}; }}}}}
```

SPE10 Test Case

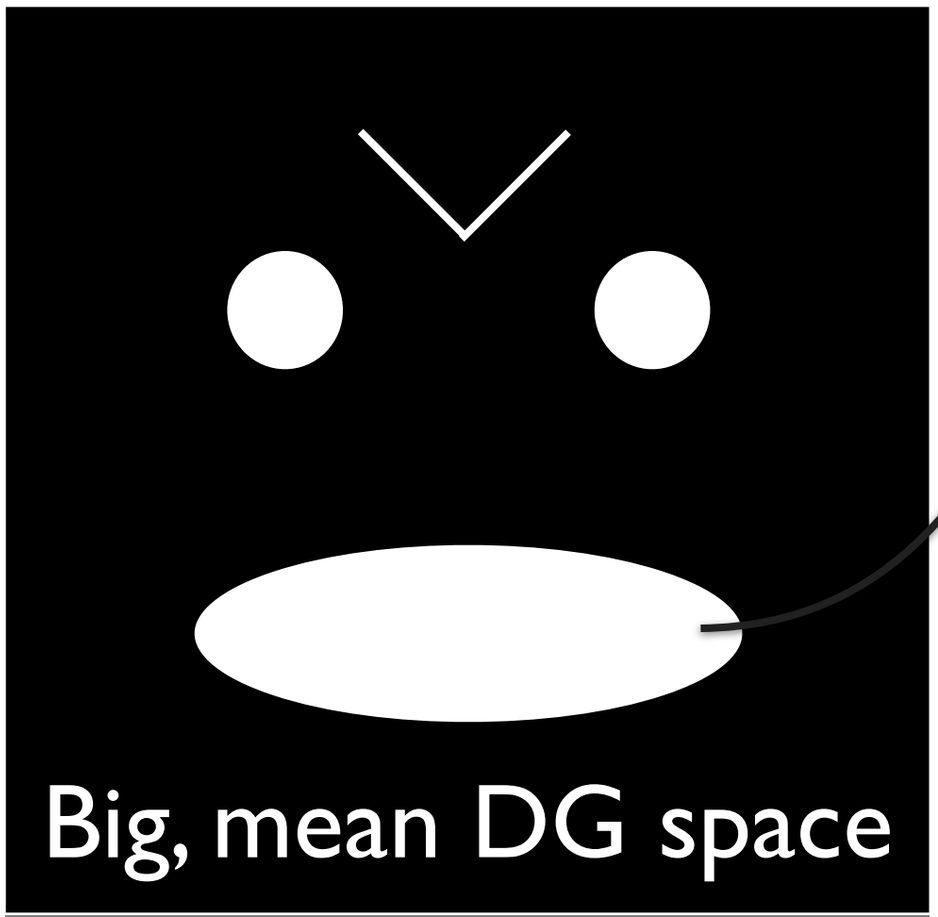
- Multiscale problem: Huge domain, highly heterogenous permeability, mobility changes on large scale with parameter
- Problem: Computing too many detailed simulations unfeasible (offline time: 6 hours)
- Localized Reduced Basis Multiscale method: Combine RB method with domain decomposition

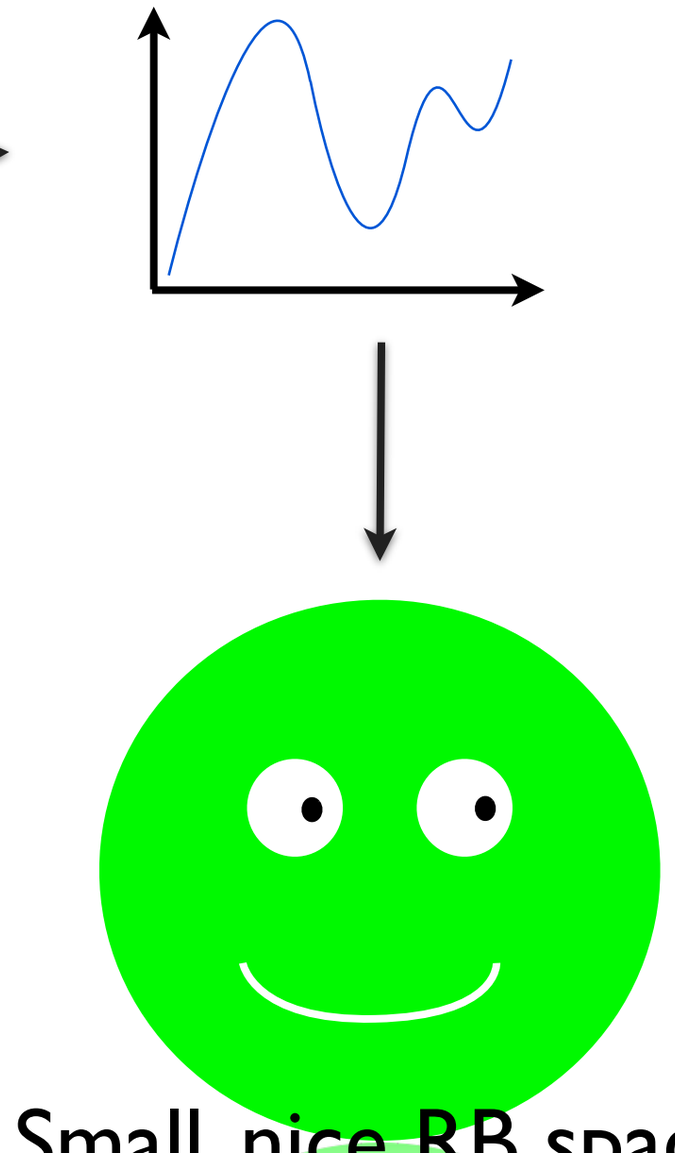


- S. Kaulmann, M. Ohlberger and B. Haasdonk: *A new local reduced basis discontinuous galerkin approach for heterogeneous multiscale problems*. Comptes Rendus Mathematique, 349(23-24):1233–1238, December 2011
- F. Albrecht, B. Haasdonk, S. Kaulmann and M. Ohlberger: *The Localized Reduced Basis Multiscale Method*. SimTech Preprint 3/2012

Localized Multiscale RB Method

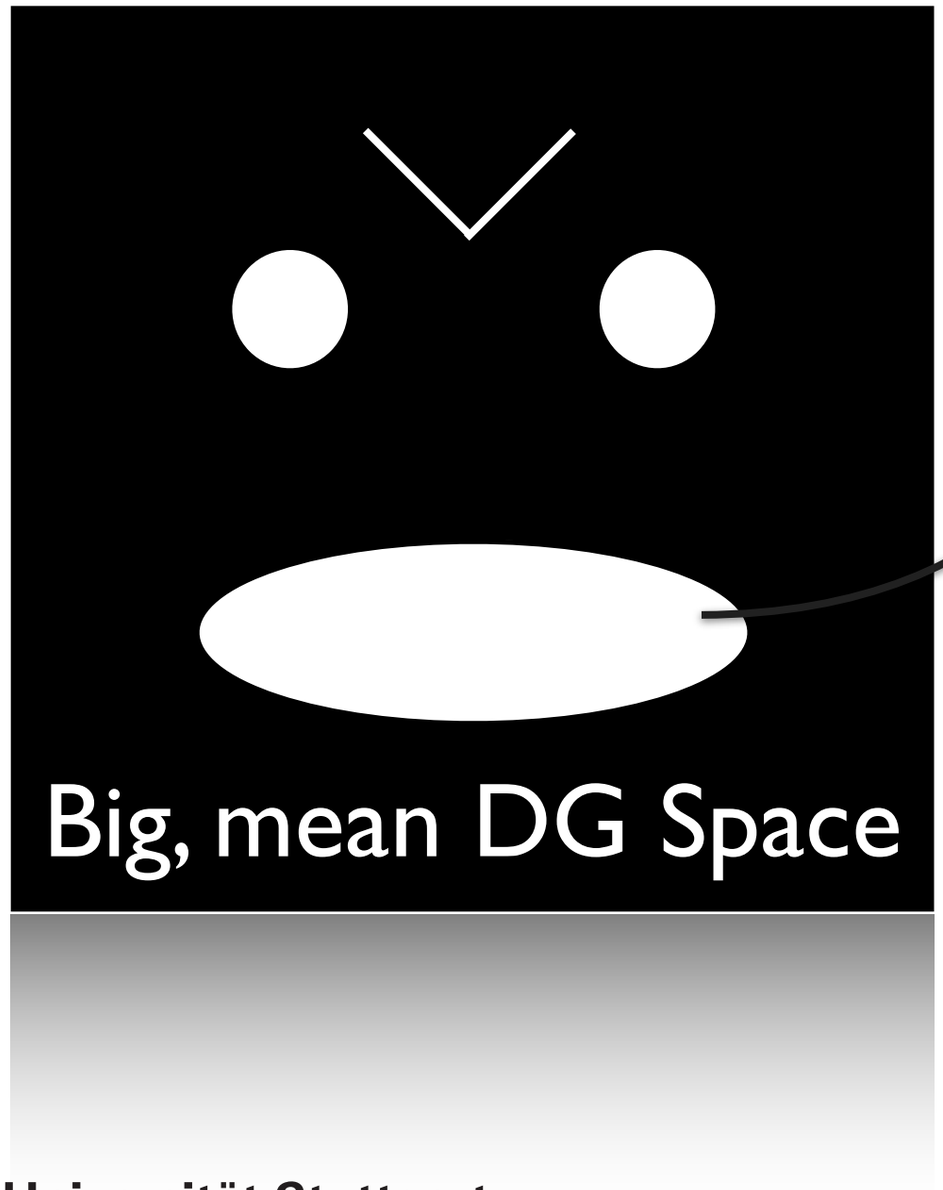
So far:

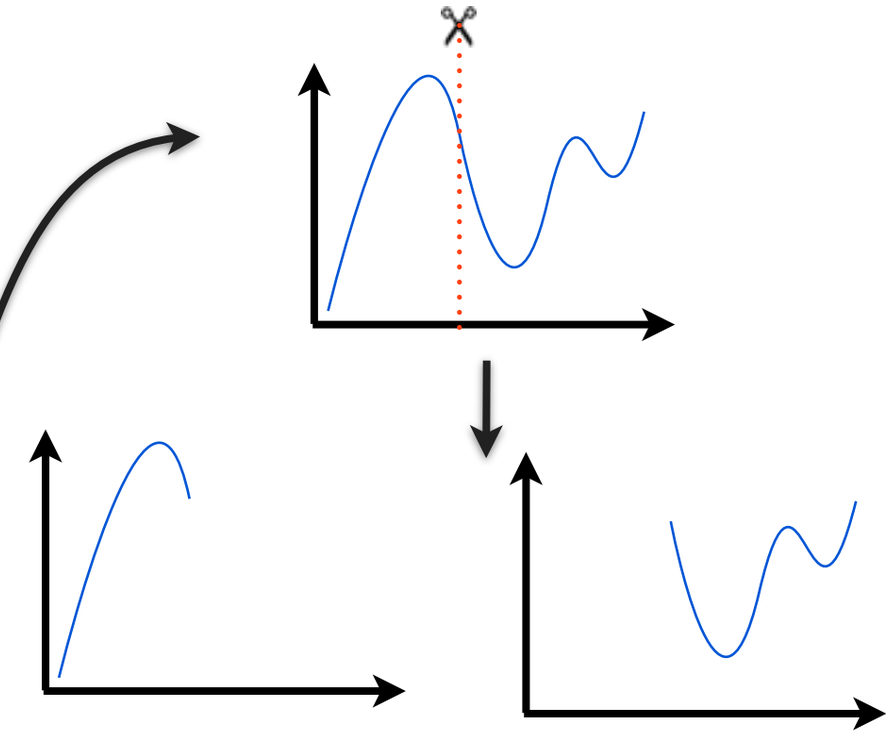




Localized Multiscale RB Method

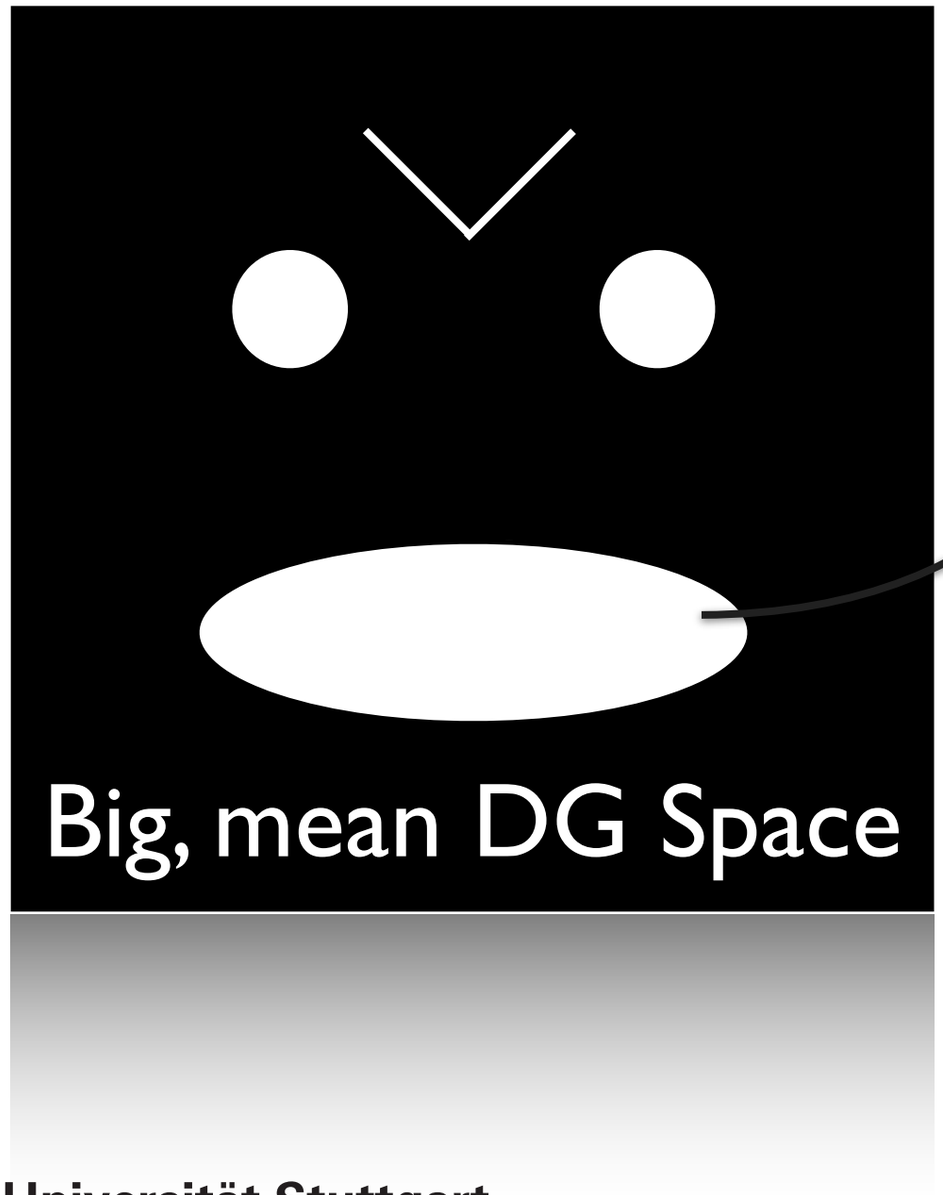
Now:

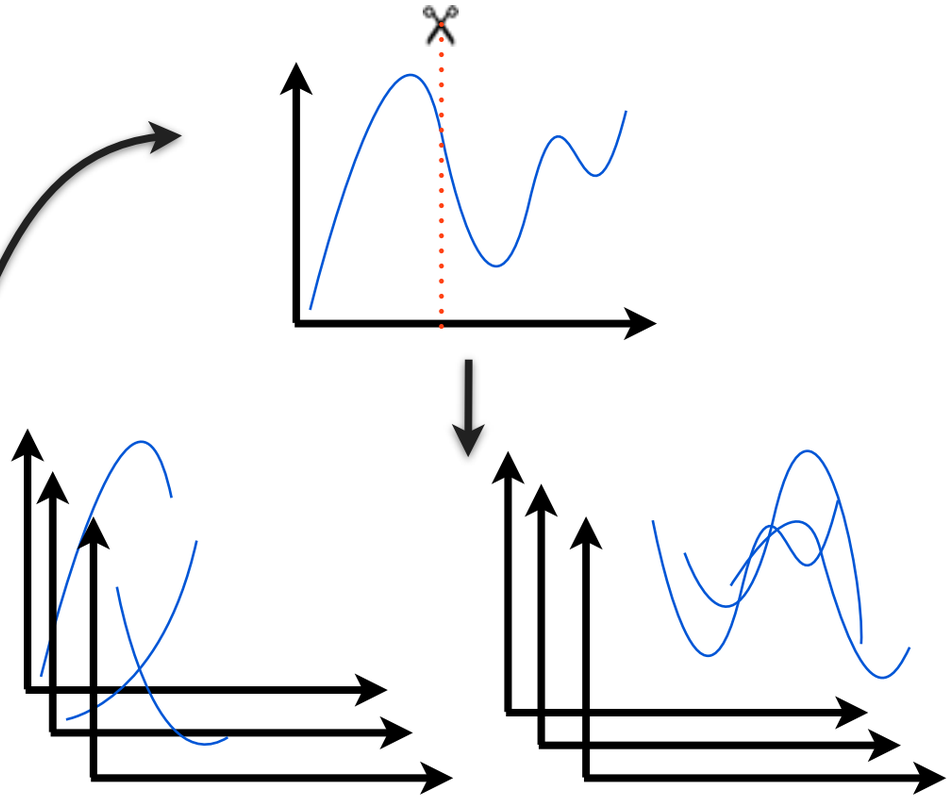




Localized Multiscale RB Method

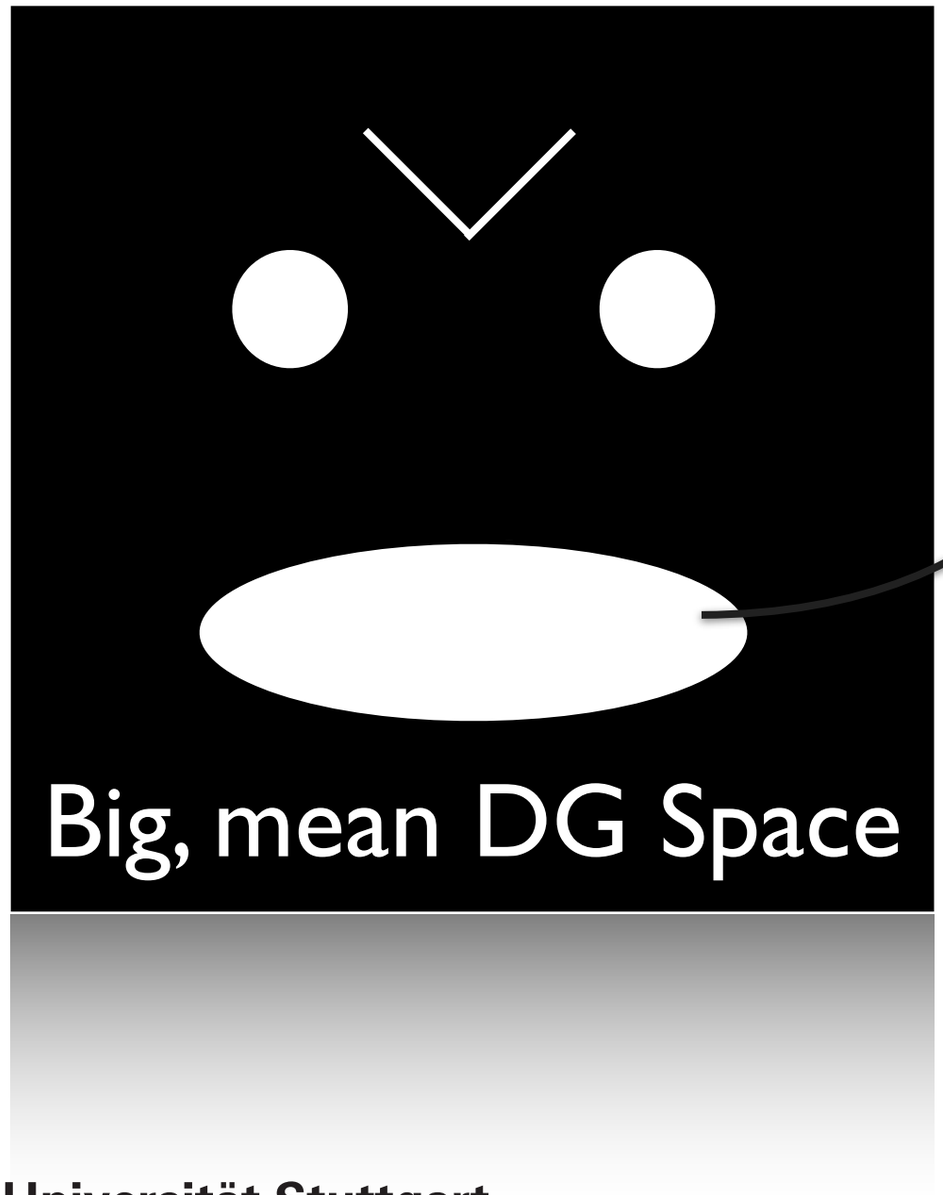
Now:

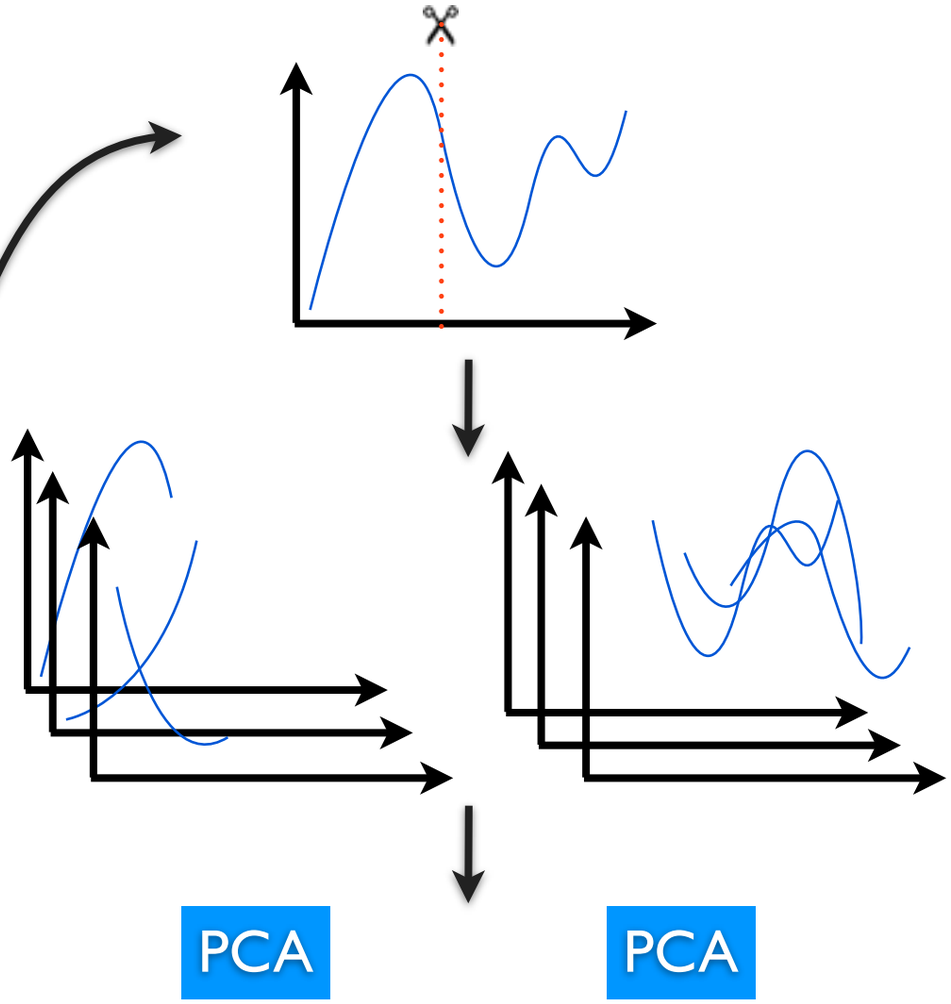




Localized Multiscale RB Method

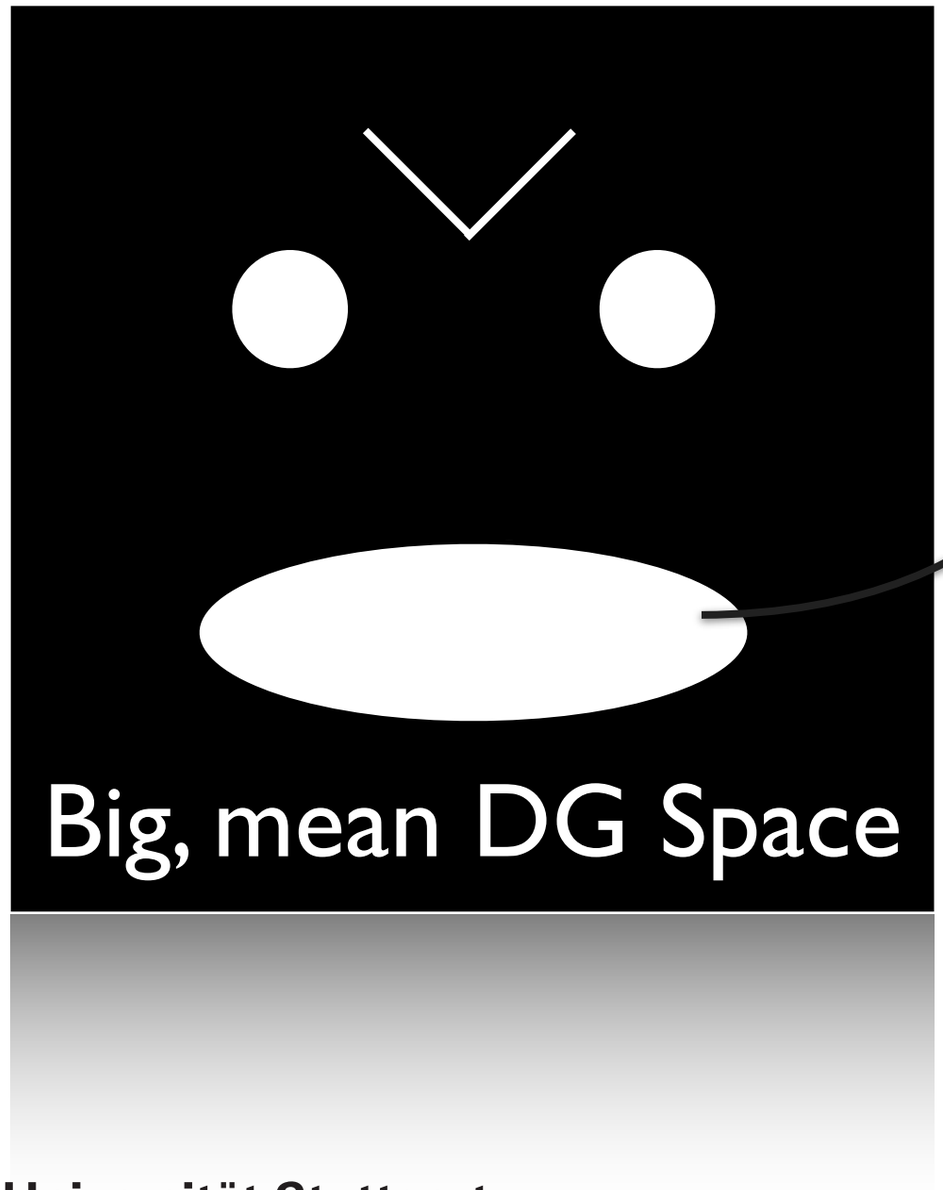
Now:

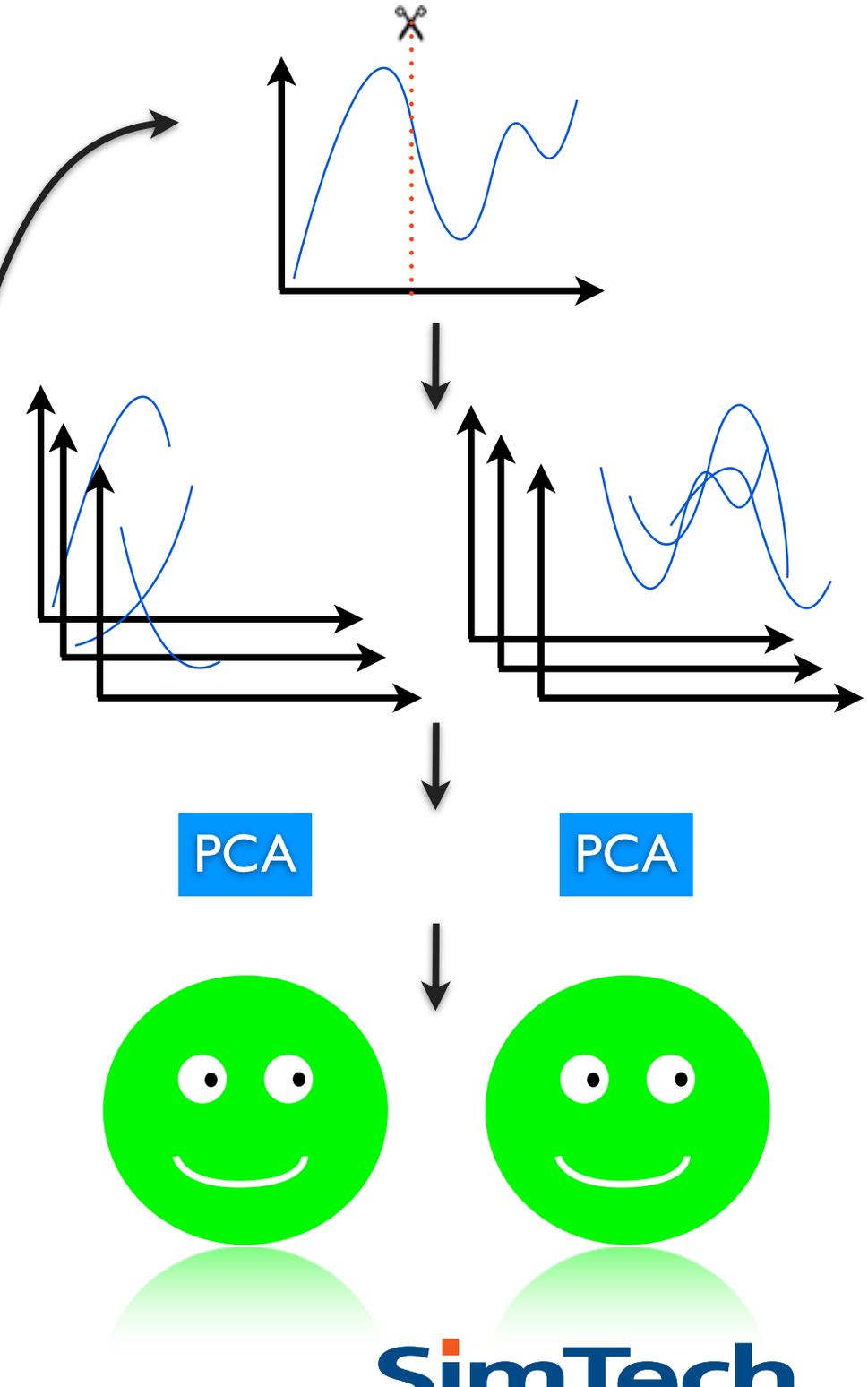




Localized Multiscale RB Method

Now:



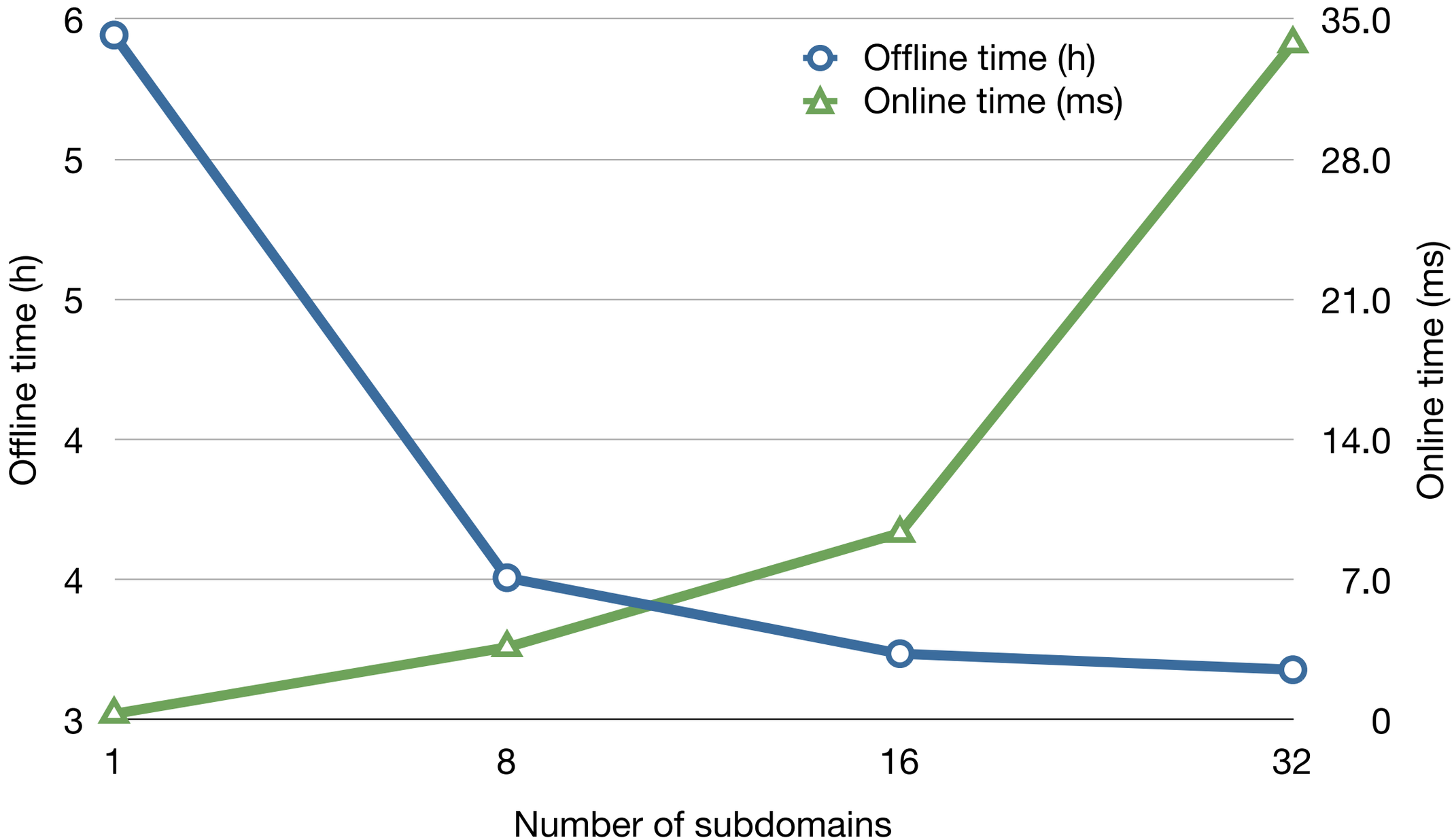


Implementation of LMRB Method

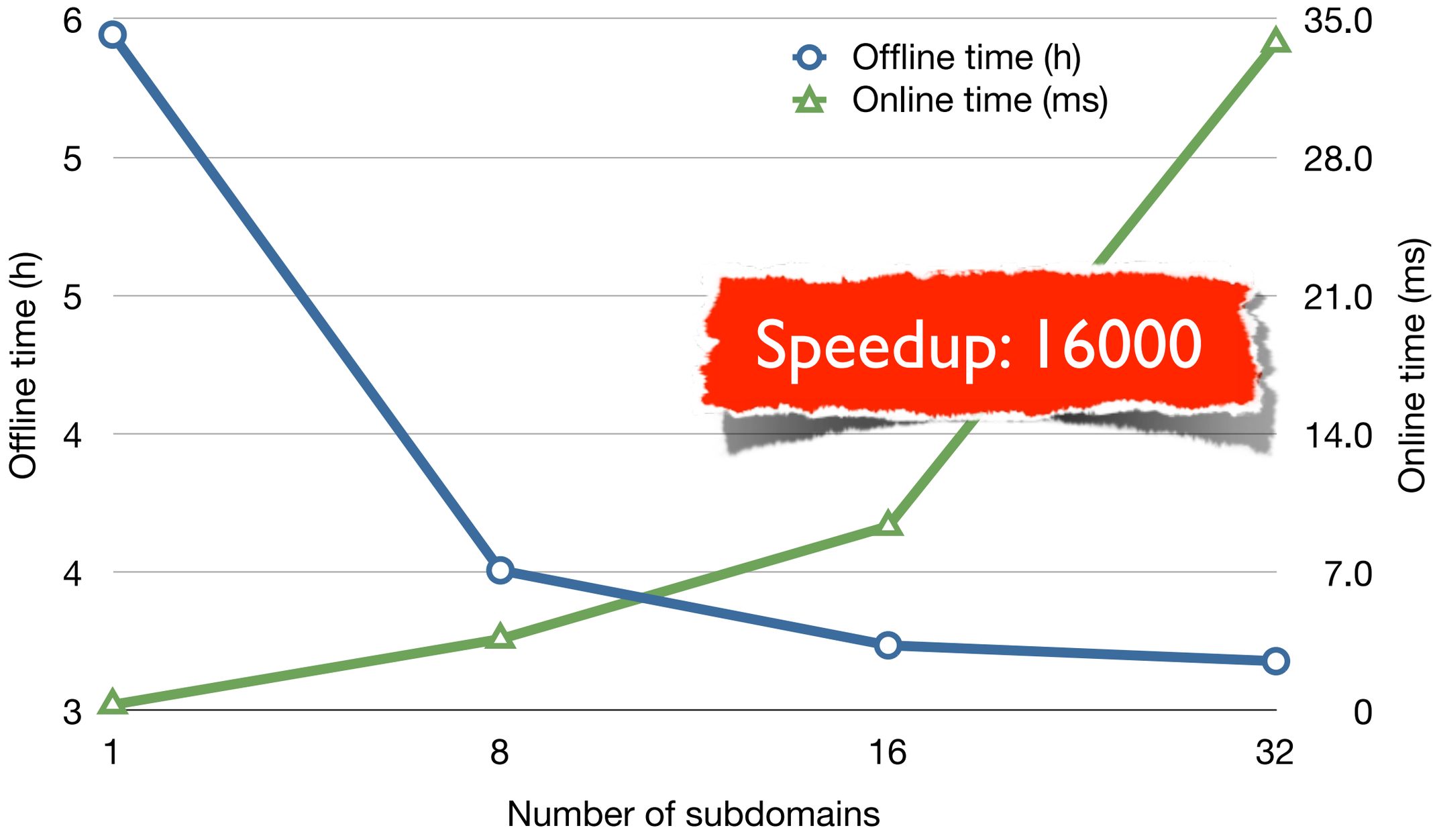
Additional challenges:

- Decomposition of spatial grid
  - Index based division of grid
  - Parser for dgf-style subdivision files for 2D & 3D
- Localization of functions
  - Restrict DOF vector based on index set built with detailed mapper
  - Possible because RB space does not rely on “functions“
- Principal Component Analysis
  - Easy (solutions as DOF vectors, usage of Eigen)

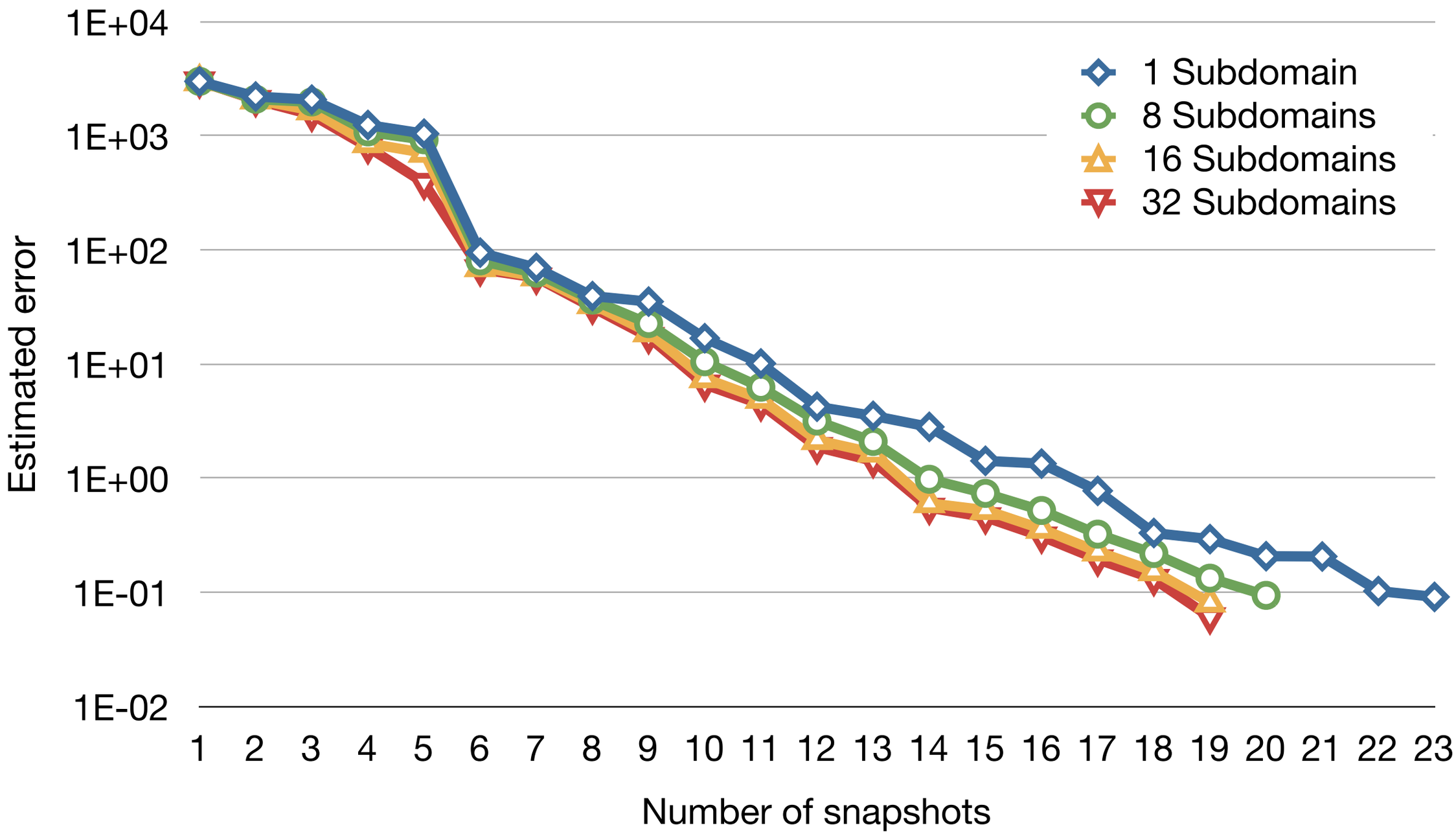
SPE10 Offline vs. Online Time



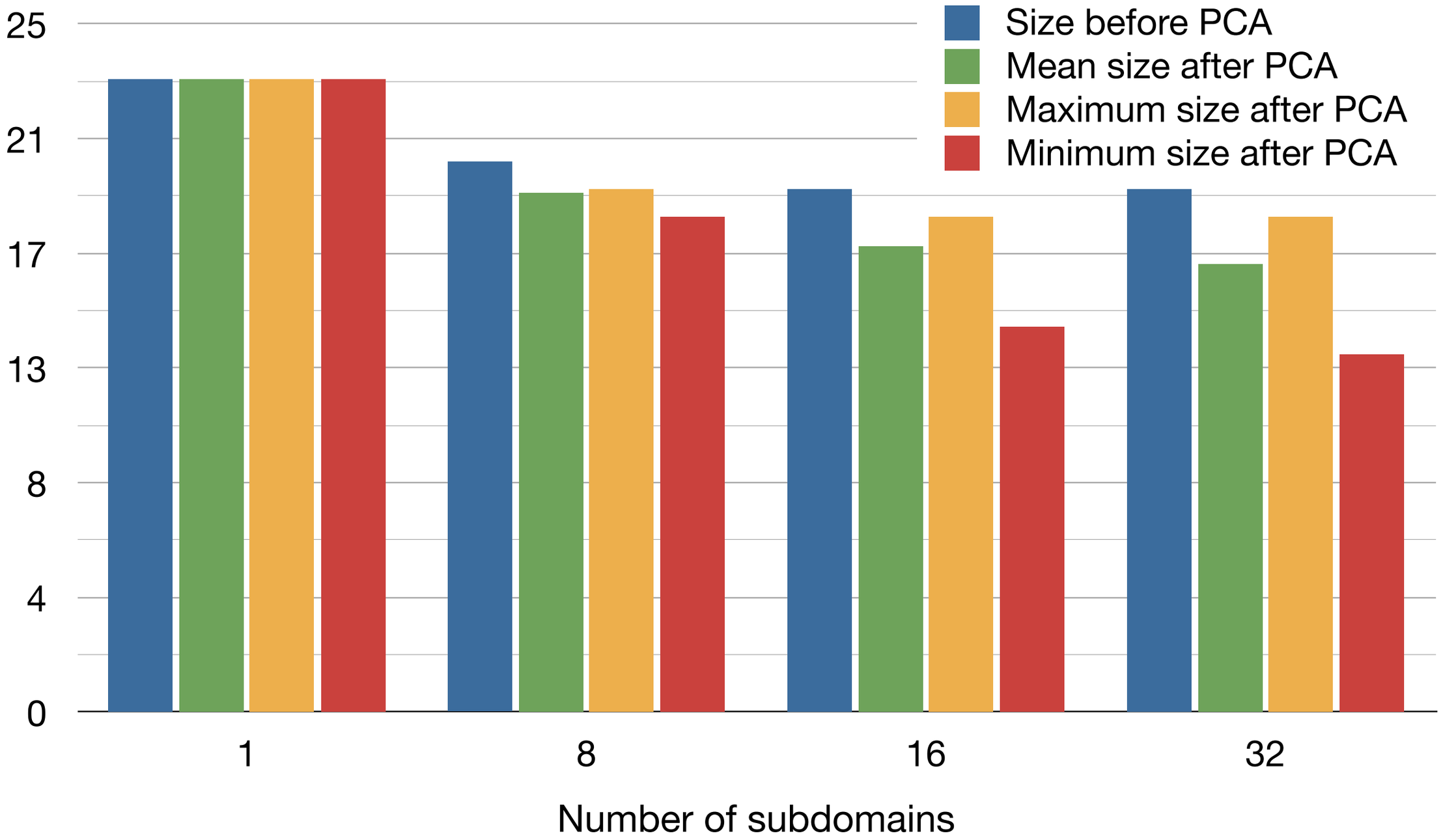
SPE10 Offline vs. Online Time



SPE10 Error Convergence



SPE10 PCA


25
21
17
13
8
4
0
1
8
16
32
Number of subdomains
Size before PCA
Mean size after PCA
Maximum size after PCA
Minimum size after PCA

Outlook: Python-prototyping

- Expose C++ code to python using boost.python
- All your computationally expensive tasks implemented in C++ but used as python module
- Easy prototyping for all reduced problems
- GUI implementation in “no time“

Outlook: Python-prototyping

C++

```
#include <iostream>
#include <boost/python.hpp>

struct Calculator {

  Calculator() {}

  double sum(double a, double b) const
  {
    return a+b;
  }

  double product(double a, double b) const
  {
    return a*b;
  }

  void print(std::string message) const
  {
    std::cout << message;
  }
};

BOOST_PYTHON_MODULE(MyModule)
{
  boost::python::class_<Calculator>("Calculator")
    .def("sum", &Calculator::sum)
    .def("product", &Calculator::product)
    .def("printMessage", &Calculator::print)
  ;
}
```

Outlook: Python-prototyping

C++

```cpp
#include <iostream>
#include <boost/python.hpp>

struct Calculator {

  Calculator() {}

  double sum(double a, double b) const
  {
    return a+b;
  }

  double product(double a, double b) const
  {
    return a*b;
  }

  void print(std::string message) const
  {
    std::cout << message;
  }
};

BOOST_PYTHON_MODULE(MyModule)
{
  boost::python::class_<Calculator>("Calculator")
    .def("sum", &Calculator::sum)
    .def("product", &Calculator::product)
    .def("printMessage", &Calculator::print)
  ;
}
```

Python

```python
from distutils.core import setup, Extension

myModule = Extension('MyModule',
    include_dirs = ['/opt/local/include'],
    libraries = ['boost_python'],
    library_dirs = ['/opt/local/lib'],
    sources = ['calculator.cc'])

setup (name = 'MyModule',
       version = '0.1',
       description = 'Awesome!',
       ext_modules = [myModule])
```

Outlook: Python-prototyping

C++

```cpp
#include <iostream>
#include <boost/python.hpp>

struct Calculator {

  Calculator() {}

  double sum(double a, double b) const
  {
    return a+b;
  }

  double product(double a, double b) const
  {
    return a*b;
  }

  void print(std::string message) const
  {
    std::cout << message;
  }
};

BOOST_PYTHON_MODULE(MyModule)
{
  boost::python::class_<Calculator>("Calculator")
    .def("sum", &Calculator::sum)
    .def("product", &Calculator::product)
    .def("printMessage", &Calculator::print)
  ;
}
```

Python

```python
from distutils.core import setup, Extension

myModule = Extension('MyModule',
    include_dirs = ['/opt/local/include'],
    libraries = ['boost_python'],
    library_dirs = ['/opt/local/lib'],
    sources = ['calculator.cc'])

setup (name = 'MyModule',
       version = '0.1',
       description = 'Awesome!',
       ext_modules = [myModule])
```

Python

```python
from MyModule import *
calc = Calculator()
print 'Sum:', calc.sum(1.2, 4.3)
print 'Product:', calc.product(1.2, 4.3)
calc.printMessage('PDESoft 2012 rules!\n')
```

Outlook: Python-prototyping

C++

```cpp
#include <iostream>
#include <boost/python.hpp>

struct Calculator {

  Calculator() {}

  double sum(double a, double b) const
  {
    return a+b;
  }

  double product(double a, double b) const
  {
    return a*b;
  }

  void print(std::string message) const
  {
    std::cout << message;
  }
};

BOOST_PYTHON_MODULE(MyModule)
{
  boost::python::class_<Calculator>("Calculator")
    .def("sum", &Calculator::sum)
    .def("product", &Calculator::product)
    .def("printMessage", &Calculator::print)
  ;
}
```

Python

```python
from distutils.core import setup, Extension

myModule = Extension('MyModule',
    include_dirs = ['/opt/local/include'],
    libraries = ['boost_python'],
    library_dirs = ['/opt/local/lib'],
    sources = ['calculator.cc'])

setup (name = 'MyModule',
       version = '0.1',
       description = 'Awesome!',
       ext_modules = [myModule])
```

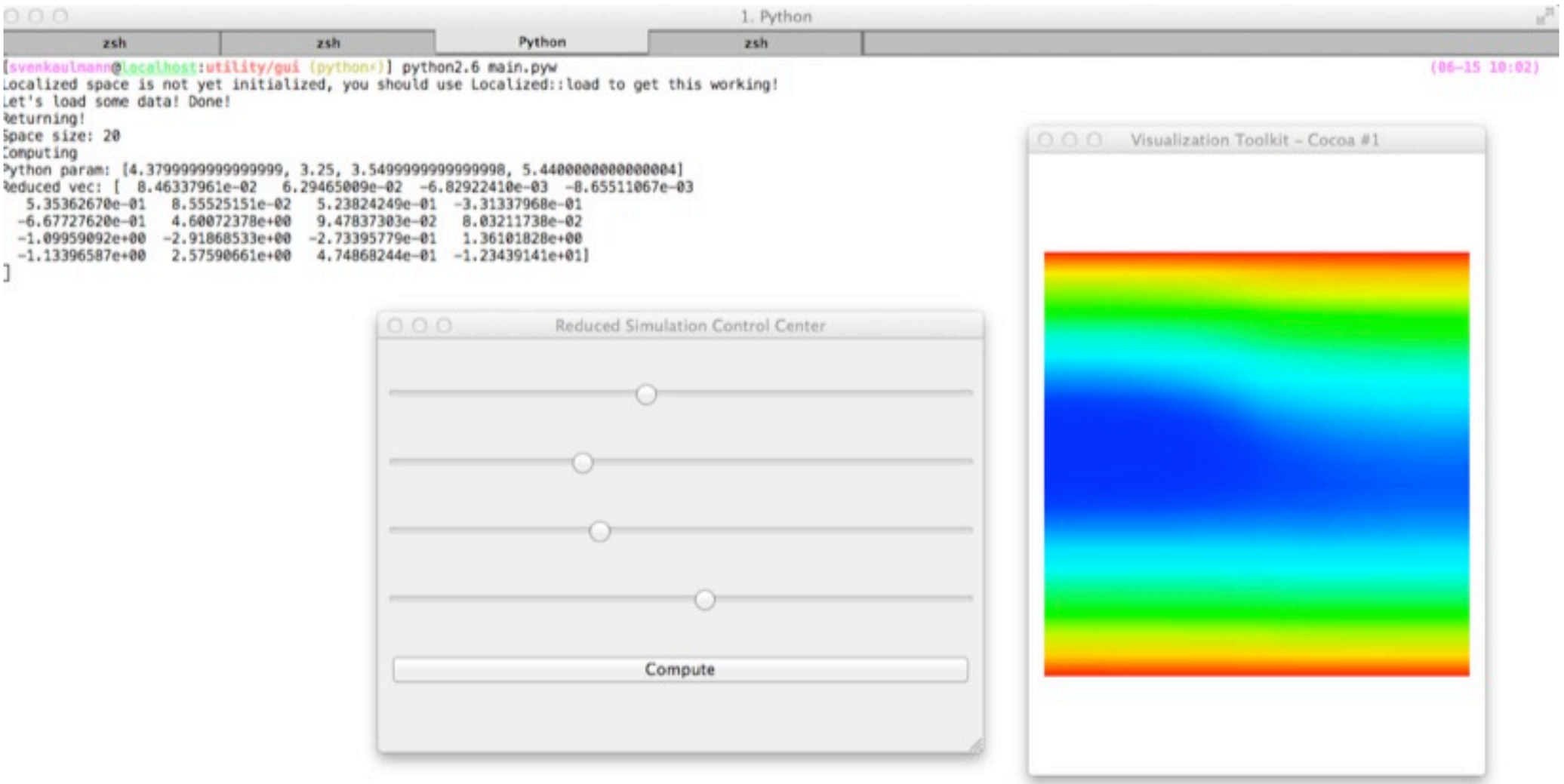
Python

```python
from MyModule import *
calc = Calculator()
print 'Sum:', calc.sum(1.2, 4.3)
print 'Product:', calc.product(1.2, 4.3)
calc.printMessage('PDESoft 2012 rules!\n')
```

Terminal

```
bash-3.2$ python2.6 runlivedemo.py
Sum: 5.5
Product: 5.16
PDESoft 2012 rules!
```

Live Demo


1. Python
zsh
zsh
Python
zsh
[svenkaulmann@localhost:utility/gui (python)] python2.6 main.pyw
(06-15 10:02)
Localized space is not yet initialized, you should use Localized::load to get this working!
Let's load some data! Done!
Returning!
Space size: 20
Computing
Python param: [4.3799999999999999, 3.25, 3.5499999999999998, 5.4400000000000004]
Reduced vec: [8.46337961e-02 6.29465009e-02 -6.82922410e-03 -8.65511067e-03
5.35362670e-01 8.55525151e-02 5.23824249e-01 -3.31337968e-01
-6.67727620e-01 4.60072378e+00 9.47837303e-02 8.03211738e-02
-1.09959092e+00 -2.91868533e+00 -2.73395779e-01 1.36101828e+00
-1.13396587e+00 2.57590661e+00 4.74868244e-01 -1.23439141e+01]
]
Reduced Simulation Control Center
Compute
Visualization Toolkit - Cocoa #1

SimTech
Cluster of Excellence

Universität Stuttgart

Visit us on www.morepas.org

SimTech
Cluster of Excellence

Universität Stuttgart

Visit us on www.morepas.org

Thank you for your attention!