Feel++ Opus : an open computational framework for reduced basis methods a SCM contribution and some applications

S. Vallaghe

Laboratoire Jean Kuntzmann – EDP
Université de Grenoble

Paris
June 23,24 2011

Collaborators

C. Prud'homme (UdG/LJK)

M. Fouquembergh(EADS-IW)

S. Vallaghe (UdG/LJK)

A. Le-Hyaric(EADS-IW)

O. Souhar

S. Veys (UdG/LJK)

A. Samake (UdG/LJK)

Sponsors

ANR OPUS

Sommaire

1. Feel++-Opus : An open reduced basis framework
2. An improved SCM offline stage

An open reduced basis framework

Objectives

- Provide an open framework for certified reduced basis methods
- Provide a laboratory for methodology experimentation
- Provide a rapid prototyping framework using the Feel++ language for the standard finite/spectral element methods
- Provide interfaces to various open "mathematical" programming environments such as Python/OpenTURNS(UQ) or Octave

Where to get it ?

- sources are available at `http://www.opus-project.fr`
- available as Debian packages for amd64 at `http://debian.feelpp.org`

Feel++

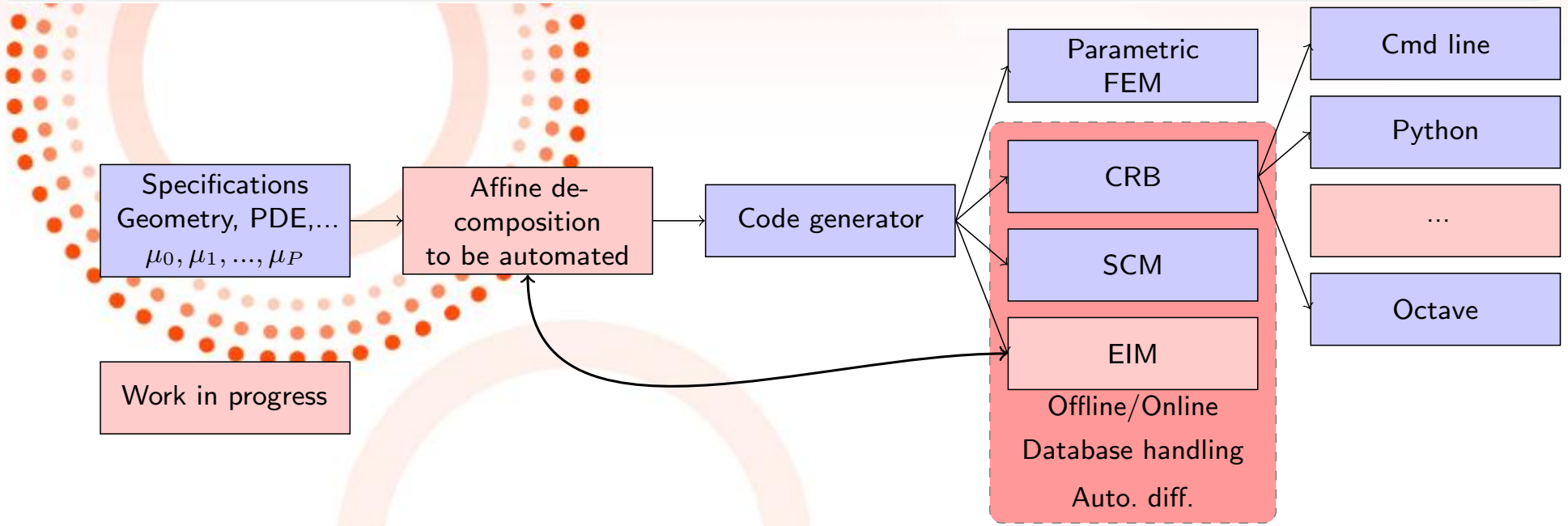
Features

- Galerkin methods (fem,sem, cG, dG) in 1D, 2D and 3D on simplices and hypercubes
- Interfaces to PETSc/SLEPc and Trilinos
- Language embedded in C++ close to variational formulation language that shortens tremendously the "time to results"

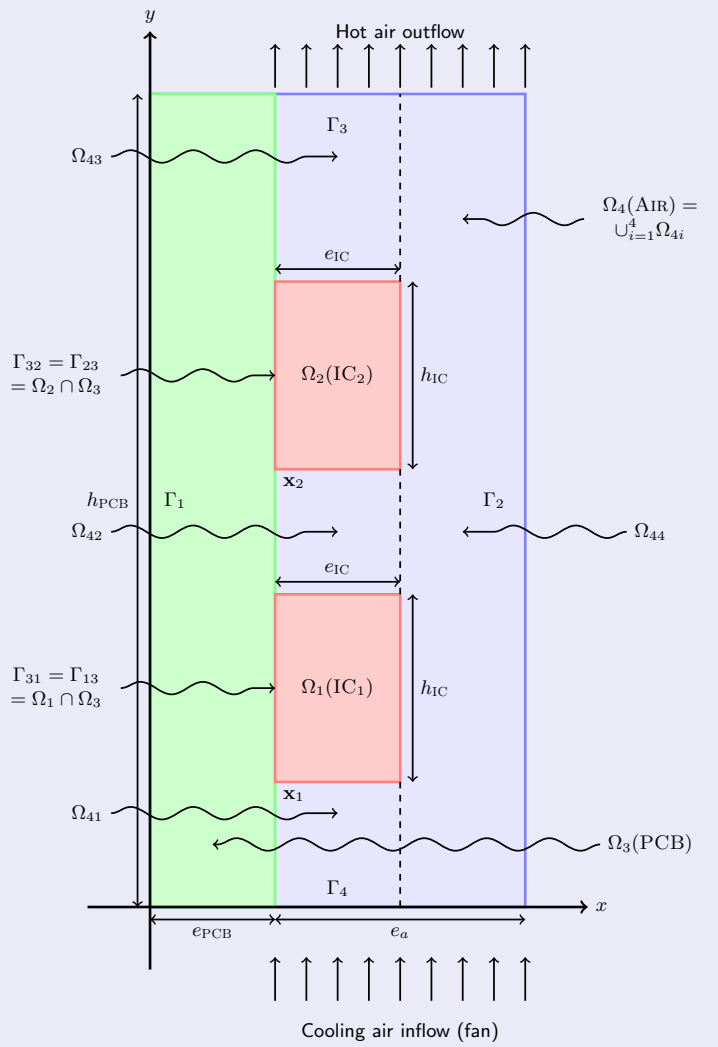
```
auto mesh = loadGmshMesh( ... );
auto Xh = FunctionSpace<Mesh<Simplex<3>>,
                        Lagrange<8,Scalar>> : :New(mesh);
auto u = Xh->element(), v = Xh->element();
form2( _test=Xh, _trial=Xh, _matrix=S ) =
  integrate( elements(mesh),  gradt(u)*trans(grad(v)));
form2( _test=Xh, _trial=Xh, _matrix=M ) =
  integrate( elements(mesh),  idt(u)*id(v) );
auto modes = eigs( _matrixA=S, _matrixB=M,
                   [options to control SLEPc eigensolver] );
```

Feel++ Opus Features

- CRB with some sampling strategies for RB generation(Greedy is the default)
- SCM implementations for coercivity or inf-sup lower/upper bounds. [Huynh et al., 2007, Chen et al., 2008, Huynh et al., 2010, Vallaghe et al., 2011]
- Empirical interpolation even if affine [Barrault et al., 2004, Eftang et al., 2010]
- Online stage automatic differentiation (not numerical) with respect to μ (gradient and hessian)



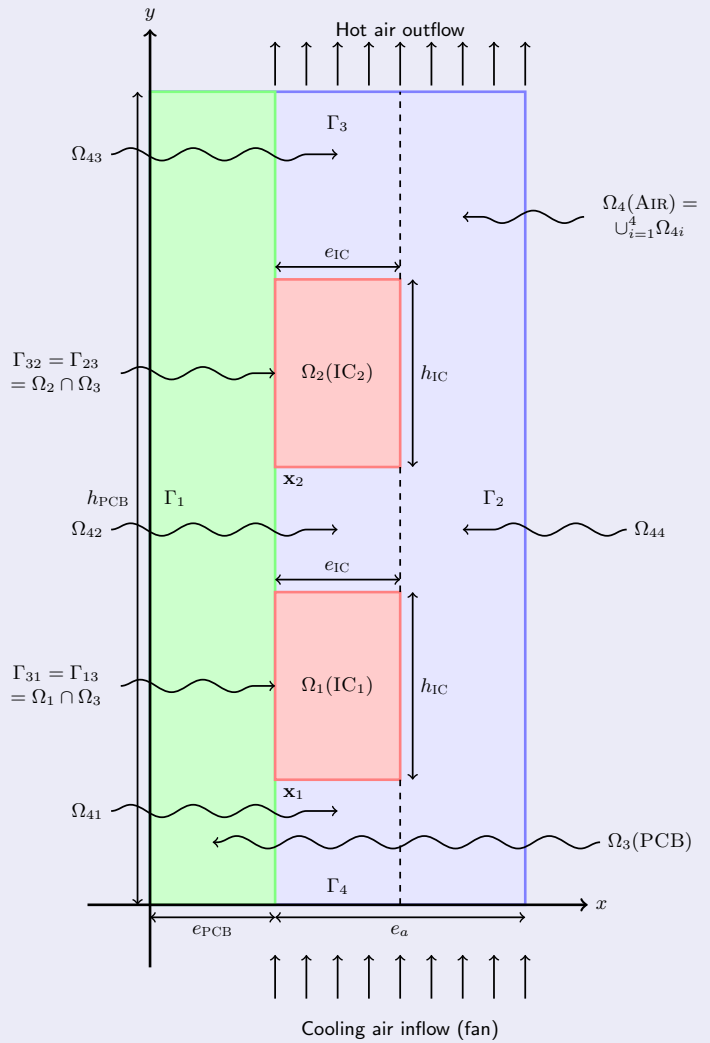
Example : Cooling of electronic components



Overview

- Heat-Transfer with conduction and convection possibly coupled with Poiseuille/Navier-Stokes
- Not a real industrial application but close enough and contains all difficulties to test the certified reduced basis
  - non symmetric, non compliant
  - steady/unsteady
  - physical and geometrical parameters
  - coupled models
- Implementation Feel++, Comsol
- Validation/Comparison between both implementations (slight differences : BC, stabilisation, meshes, discontinuities...)

Example : Cooling of electronic components



Heat transfer equation

$$\rho C_i \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) - \nabla \cdot (k_i \nabla T) = Q_i, \quad i = 1, 2, 3, 4 \quad (1)$$

Inputs/Outputs

- Five parameters

$$\nu = \{e_a; k_{Ic}; D; Q; r\}. \quad (2)$$

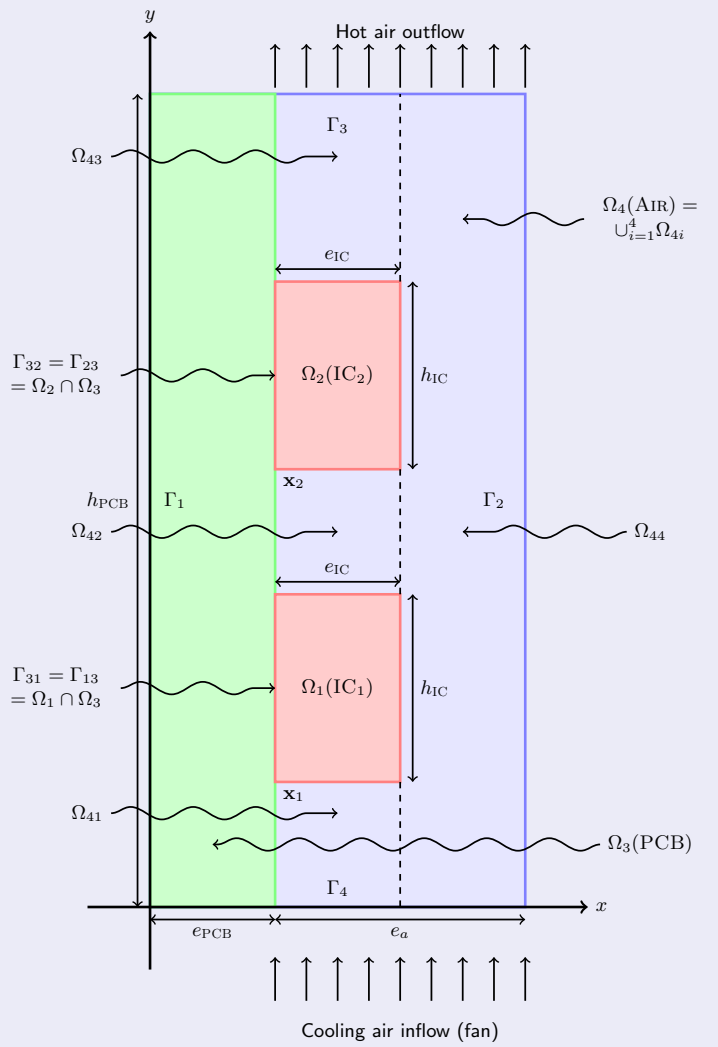
- The mean temperature $s_1(\nu)$ of the hottest IC

$$s_1(\nu) = \frac{1}{e_{IC} h_{IC}} \int_{\Omega_2} T \quad (3)$$

- The mean temperature $s_2(\nu)$ of the air at the outlet

$$s_2(\nu) = \frac{1}{e_a} \int_{\Omega_4 \cap \Gamma_3} T \quad (4)$$

Example : Cooling of electronic components



Python Code using OpenTURNS wrapper

```
# load finite element approximation
eadspfem =
 NumericalMathFunction("eadspfem")
# load reduced basis approximation
eadscrb =
 NumericalMathFunction("eadscrb")

mu[0] = 10    # kIC
mu[1] = 7e-3  # D
mu[2] = 1e6   # Q
mu[3] = 100   # r
mu[4] = 4e-3  # ea

print "mu=",mu
sfem = eadspfem(mu)
scrb = eadscrb(mu)
```

On-going work and perspectives

On going work

- Integration of EIM (at the moment separate from CRB)
- Seamless online stage automatic differentiation
- Time-dependent CRB
- Natural norm SCM

Perspectives

- More methods and applications : Reduced basis element method, reduced basis multiscale methods, reduced basis in UQ context...
- More applications
- More people using/developing the framework. You are most welcome to participate and/or give feedback

Sommaire

1. Feel++-Opus : An open reduced basis framework
2. An improved SCM offline stage

SCM [Huynh et al., 2007, Chen et al., 2008]

Affine bilinear form

$$a(w,v;\mu) = \sum_{q=1}^{Q} \theta_q(\mu)\, a_q(w,v), \quad w,v \in X^{\mathcal{N}}, \qquad \mu \in \mathcal{D}$$

Coercivity constant

$$\alpha^{\mathcal{N}}(\mu) = \inf_{y\in\mathcal{Y}} \mathcal{J}^{\text{obj}}(\mu;y) \quad \text{where} \quad \mathcal{J}^{\text{obj}}(\mu;y) \equiv \sum_{q=1}^{Q} \theta_q(\mu) y_q$$

$$\mathcal{Y} = \left\{ y \in \mathbb{R}^Q \,|\, \exists w \in X^{\mathcal{N}} \text{ s.t. } y_q = \frac{a_q(w,w;\mu)}{\|w\|^2_{X^{\mathcal{N}}}}, 1 \le q \le Q \right\}$$

Lower and upper bounds for $\alpha^{\mathcal{N}}(\mu)$

Build $\mathcal{Y}_{\text{UB}} \subset \mathcal{Y} \subset \mathcal{Y}_{\text{LB}}$ over which minimisation is feasible.

SCM [Huynh et al., 2007, Chen et al., 2008]

Bounds for the minimisation problem

$$\mathcal{B} = \prod_{q=1}^{Q} \left[\inf_{w \in X^{\mathcal{N}}} \frac{a_q(w,w;\mu)}{\|w\|_X^2} ; \sup_{w \in X^{\mathcal{N}}} \frac{a_q(w,w;\mu)}{\|w\|_X^2} \right]$$

Samplings of the parameter space

- Fine sampling $\Xi = \left\{ \mu_i \in \mathcal{D}; i = 1, ..., J \right\}$.
- Well chosen sub-sampling : $C_K = \left\{ \mu_k \in \Xi; k = 1, ..., K \right\} \subset \Xi$ (will be constructed using a greedy algorithm). Compute $\alpha^{\mathcal{N}}(\mu_k)$.

α_{UB}

- $\mathcal{Y}_{\mathrm{UB}}(C_K) = \left\{ y^*(\mu_k), 1 \leq k \leq K \right\} \quad y^*(\nu) = \mathrm{argmin}_{y \in \mathcal{Y}} \, \mathcal{J}^{\mathrm{obj}}(\nu; y)$
- $\alpha_{\mathrm{UB}}(\nu; C_K) = \inf_{y \in \mathcal{Y}_{\mathrm{UB}}(C_K)} \mathcal{J}^{\mathrm{obj}}(\nu; y)$

SCM [Huynh et al., 2007, Chen et al., 2008]

Coercivity constant definition

α_{LB}

$$\mathcal{Y}_{\mathrm{LB}}(\nu; C_K) = \Big\{ y \in \mathcal{B} \,|\, \sum_{q=1}^{Q} \theta_q(\nu') y_q \geq \alpha^{\mathcal{N}}(\nu') \,,\ \forall \nu' \in P_{M_\alpha}(\nu; C_K)$$

$$\sum_{q=1}^{Q} \theta_q(\nu') y_q \geq \alpha_{\mathrm{LB}}(\nu'; C_{K-1}) \,,\ \forall \nu' \in P_{M_+}(\nu; \Xi \backslash C_K) \Big\}$$

$$\alpha_{\mathrm{LB}}(\nu; C_K) = \inf_{y \in \mathcal{Y}_{\mathrm{LB}(\nu; C_K)}} \mathcal{J}^{\mathrm{obj}}(\nu; y)$$

Neighboring points of ν in samplings

Computation

Linear program with Q design variables, y_q, and $2Q + M_\alpha + M_+$ constraints.

SCM [Huynh et al., 2007, Chen et al., 2008]

Greedy algorithm for C_K construction (offline)

Given $\epsilon \in [0;1]$ and μ_1

While $\max_{\nu \in \Xi} \dfrac{\alpha_{\mathrm{UB}}(\nu; C_K) - \alpha_{\mathrm{LB}}(\nu; C_K)}{\alpha_{\mathrm{UB}}(\nu; C_K)} > \epsilon$

- $\mu_{K+1} = \mathrm{argmax}_{\nu \in \Xi} \dfrac{\alpha_{\mathrm{UB}}(\nu; C_K) - \alpha_{\mathrm{LB}}(\nu; C_K)}{\alpha_{\mathrm{UB}}(\nu; C_K)}$
- $C_{K+1} = C_K \cup \{\mu_{K+1}\}$, $K \leftarrow K+1$

LP for lower bound

Constraints

$$\sum_{q=1}^{Q} \theta_q(\nu') y_q \geq \alpha^{\mathcal{N}}(\nu'), \ \forall \nu' \in P_{M_\alpha}(\nu; C_K)$$

$$\sum_{q=1}^{Q} \theta_q(\nu') y_q \geq \alpha_{\mathrm{LB}}(\nu'; C_{K-1}), \ \forall \nu' \in P_{M_+}(\nu; \Xi \backslash C_K)$$

Neighboring points of ν in E : $P_M(\nu; E)$

- M_α and M_+ are user-dependent.
- Expensive computing of $P_M(\nu; E)$ if E is large.
- Neighboring points do not necessarily give the best constraints.

Getting rid of M_α and M_+

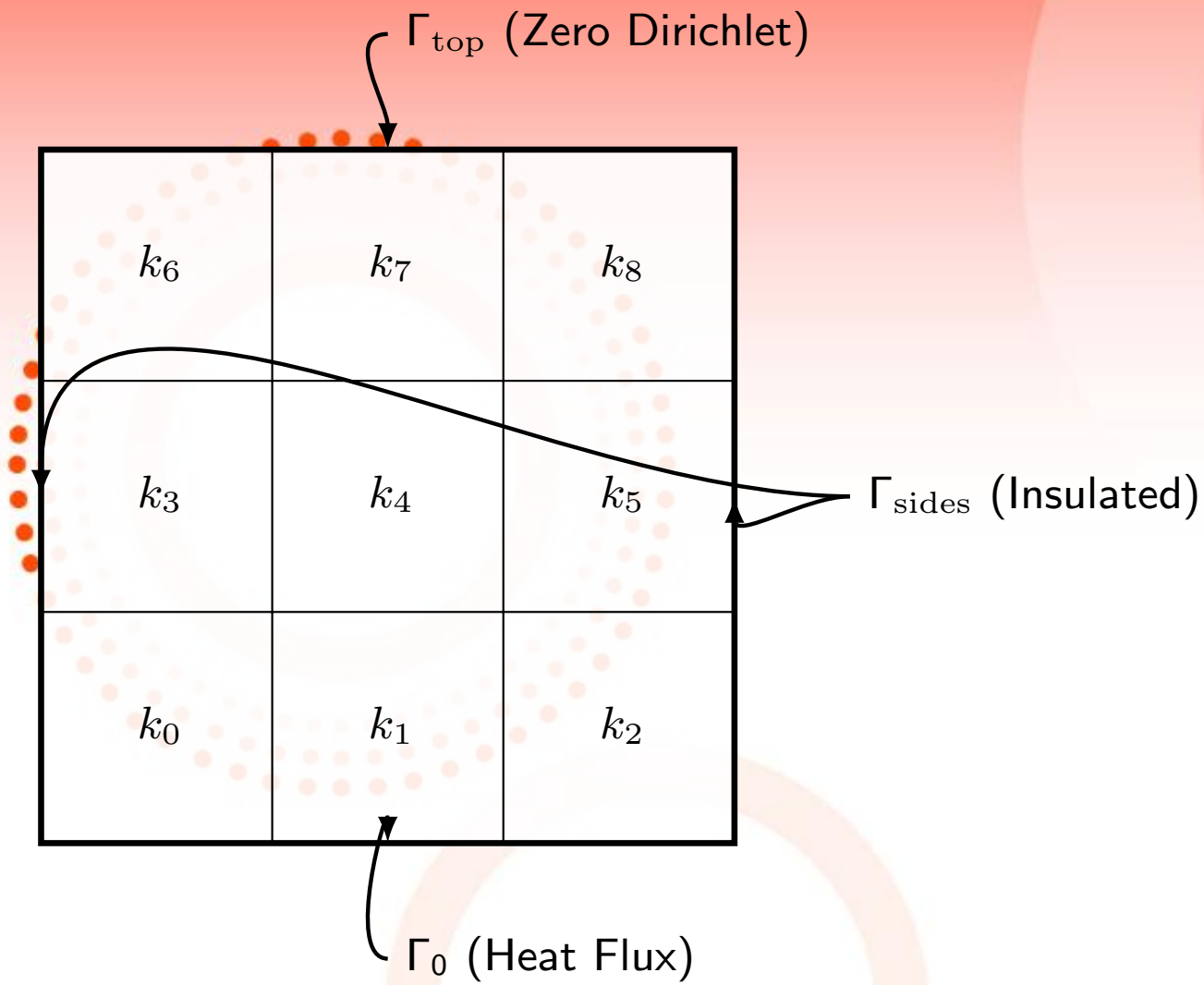
M_+

- $\alpha_{\text{LB}}(\nu'; C_{K-1})$ less sharp constraint than $\alpha^{\mathcal{N}}(\nu')$: drop M_+ constraints.
- keep only $\alpha_{\text{LB}}(\nu; C_K) \geq \alpha_{\text{LB}}(\nu; C_{K-1})$ to enforce lower bound increasing (offline).

M_α

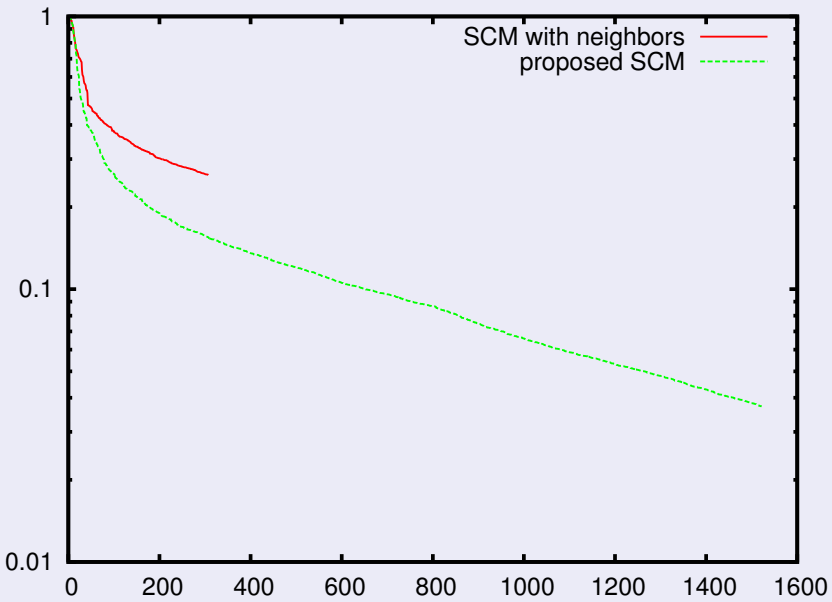
- LP with Q variables : at most Q active constraints. Take the active constraints at minimum.
- Offline : easy to update the active constraints during the construction of C_{K+1} : $\alpha^{\mathcal{N}}(\mu_{K+1})$ active ?
- Online : active constraints can't be known a priori. Take all the constraints given by $\mu \in C_K$. Many constraints if C_K is large, but considering the dual problem makes fast solving of the LP.

Example on thermalblock problem (heat transfer)



Thermalblock 3x3, Card(Ξ)=10000

$\frac{\alpha_{\mathrm{UB}} - \alpha_{\mathrm{LB}}}{\alpha_{\mathrm{UB}}}$ in the greedy algorithm



- SCM with neighbors : $M_\alpha = 10$, $M_+ = 10$.

Computational cost (offline)

- SCM with neighbors
  - Reaching $K = 300$ takes 8h.
  - Breaks after $K = 300$ because of memory load with kd-tree neighbor searching.
- Proposed SCM
  - Reaching $K = 300$ takes 9mn, $K = 1500$ takes 45mn.
  - No memory overload.

Computational cost (online)

Proposed SCM, $K = 1500$: 10^5 lower bounds computed in 1mn.

References I

Barrault, M., Nguyen, N. C., Maday, Y., and Patera, A. T. (2004).

An "empirical interpolation" method : Application to efficient reduced-basis discretization of partial differential equations.
C. R. Acad. Sci. Paris, Série I., 339 :667–672.

Chen, Y., Hesthaven, J. S., Maday, Y., and Rodríguez, J. (2008).

Improved successive constraint method based a posteriori error estimate for reduced basis approximation of 2d maxwell's problem.
Elsevier Science.
submitted.

Eftang, J. L., Grepl, M. A., and Patera, A. T. (2010).

A posteriori error bounds for the empirical interpolation method.
Comptes Rendus Mathematique, 348(9-10) :575 – 579.

Huynh, D., Knezevic, D., Chen, Y., Hesthaven, J., and Patera, A. (2010).

A natural-norm successive constraint method for inf-sup lower bounds.
Computer Methods in Applied Mechanics and Engineering, 199(29-32) :1963 – 1975.

Huynh, D., Rozza, G., Sen, S., and Patera, A. (2007).

A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants.
CR Acad Sci Paris, (345) :473–478.

Vallaghe, S., Le-Hyaric, A., Fouquemberg, M., and Prud'homme, C. (2011).

A successive constraint with minimal offline constraints.
CR Acad Sci Paris.