



**Barcelona
Supercomputing
Center**
Centro Nacional de Supercomputación

A filtering framework for Finite Volume / Element schemes

**Julia Docampo, Matthew Picklo,
Soraya Terrab and Jennifer K. Ryan**



Post-processing data with SIAC filters

① Superconvergence: for a DG/FEM solution of degree p

- Proven $2p + 1$ order (L^2 and L_∞ norms) for linear PDEs
- Observed $2p + 1$ order for non-linear PDEs

② Smoothness: the filtered data is a (local) $2p + 1$ polynomial

- Removes oscillations in the error
- Recovers continuity levels across element interfaces

Applications

Flow visualization

Shock detection

Multiresolution analysis

Cut cells



Jallepalli, Docampo, Ryan, Haines, Kirby (TVCG 2017)



Barcelona
Supercomputing
Center

Centro Nacional de Supercomputación

Post-processing data with SIAC filters

Goal:

- Establish a filtering framework for general purpose
- Create a standalone tool for general applications

➤ Overview of the SIAC kernel

Basis functions, moment preservation and superconvergence

➤ Filtering challenges

➤ The software package

Overview of the Smoothness-Increasing Accuracy-Conserving (SIAC) filter

We post-process our data via convolution:

$$\text{data}^*(x) = \int_{\mathbb{R}} K(y - x) \cdot \text{data}(y) dy$$

SIAC kernel: $K^{(r+1,n)}(\cdot) = \sum_{\gamma=1}^{r+1} c_\gamma \cdot B_{T_\gamma, n}(\cdot)$

- c : kernel **weights** chosen to maintain r moments
- $B_{T,n}$: n^{th} -order central **B-spline** with knot sequence T

SIAC kernel weights: $K^{(r+1,n)} = \sum_{\gamma=1}^{r+1} c_\gamma \cdot B_{\tau_\gamma, n}$

Filtering principle: preserve the accuracy in the data

Choose the c_γ 's to satisfy

$$\int_{\mathbb{R}} K(x) dx = 1, \quad \int_{\mathbb{R}} K(x) x^j dx = 0, \quad j = 1, 2, \dots, r$$

Consistency

+

Moment Conditions

This is equivalent to impose **polynomial reproduction**

$$\int_{\mathbb{R}} K(x - y) \cdot y^j dx = x^j, \quad j = 0, 1, \dots, r$$

To extract $2p + 1$ order, the kernel must satisfy $2p$ moments

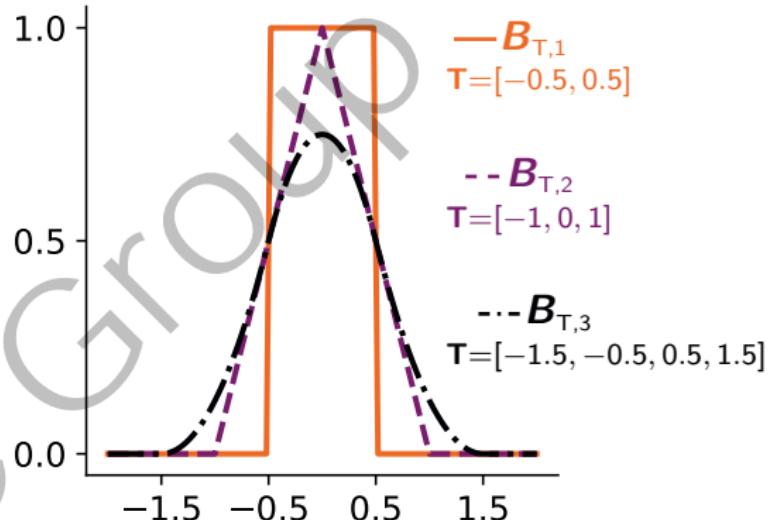
SIAC kernel basis functions: $K^{(r+1,n)} = \sum_{\gamma=1}^{r+1} c_\gamma \cdot B_{T_\gamma, n}$

B-splines

$$B_{T,1} = \chi_{[-\frac{1}{2}, \frac{1}{2}]}$$

$$B_{T,n} = B_{T,n-1} * B_{T,1}$$

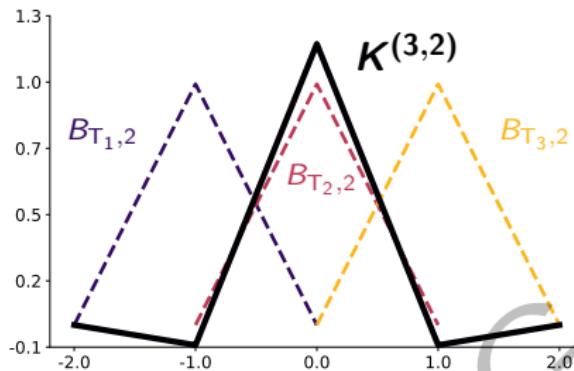
T (uniform) knot-sequence



Properties:

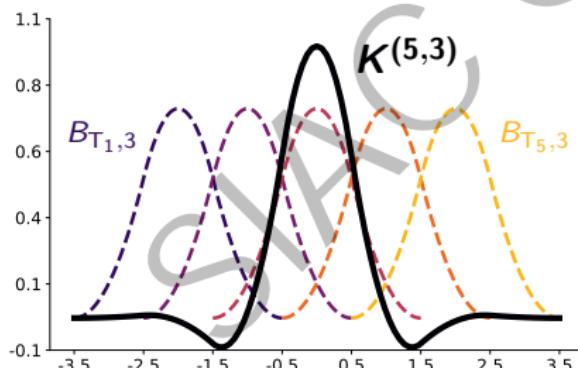
- Compact support (smaller integral region)
- Smoothness of $n - 2$ (remove oscillations)
- Derivatives as divided differences (superconvergence theory)

$$\text{SIAC Kernel: } K^{(r+1,n)}(\cdot) = \sum_{\gamma=1}^{r+1} c_\gamma B_{T_\gamma, n}(\cdot)$$



$$r = 2, n = 2$$

- Preserves 2 moments
- Continuity: \mathcal{C}^0
- Optimal order: 3



$$r = 4, n = 3$$

- Preserves 4 moments
- Continuity: \mathcal{C}^1
- Optimal order: 5

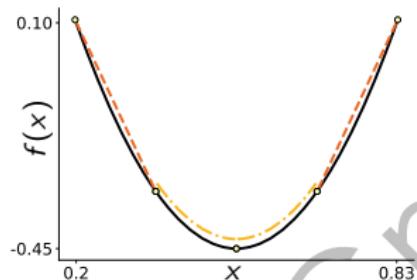
Polynomial functions: filtering linear data ($p = 1$)

Docampo, Jacobs, Li, Ryan (CAF 2020)

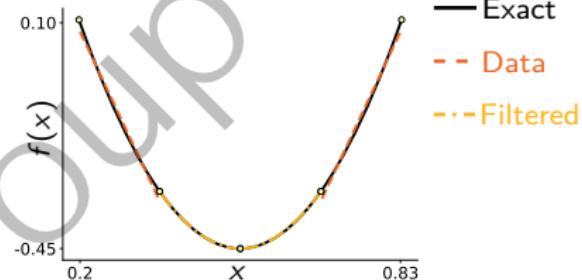
Target

Interpolant
(No Galerkin orthogonality)

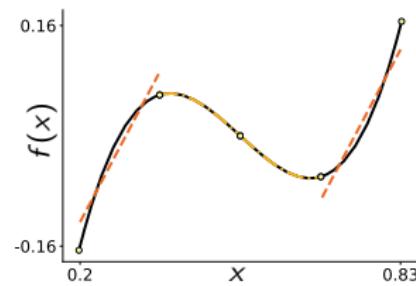
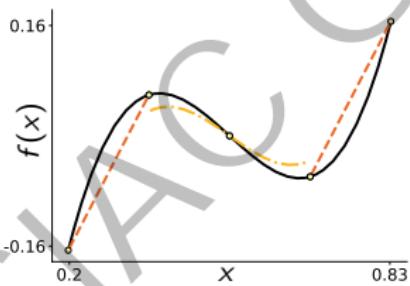
Quadratic Function



L^2 -projection

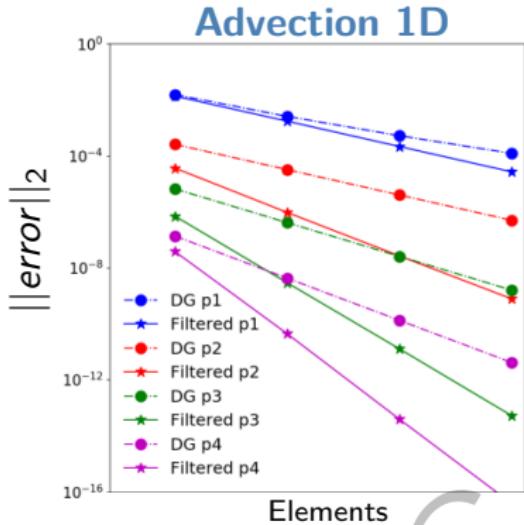


Cubic Function



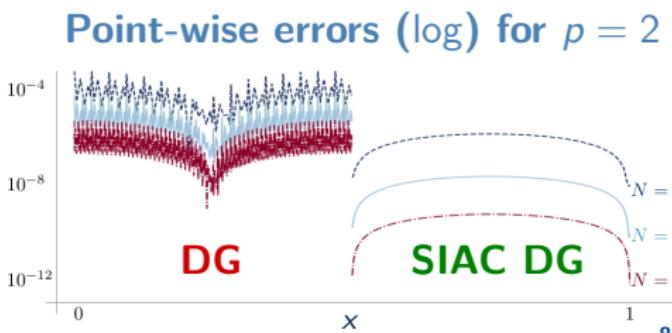
The SIAC kernel recovers the exact function for the L^2 -projected data!

SIAC filtering: superconvergence for DG solutions



→ DG approximation order: $p + 1$
→ SIAC DG order: $2p + 1$

- ✓ Reduces oscillations
- ✓ General error reduction



Post-processing data with SIAC filters

- ✓ Overview of the SIAC kernel

Moment preservation + B-splines + L^2 -initialization
⇒ **provable $2p + 1$ superconvergence**

- > Filtering challenges

Multidimensional data, domain boundaries and kernel scaling

- > The software package

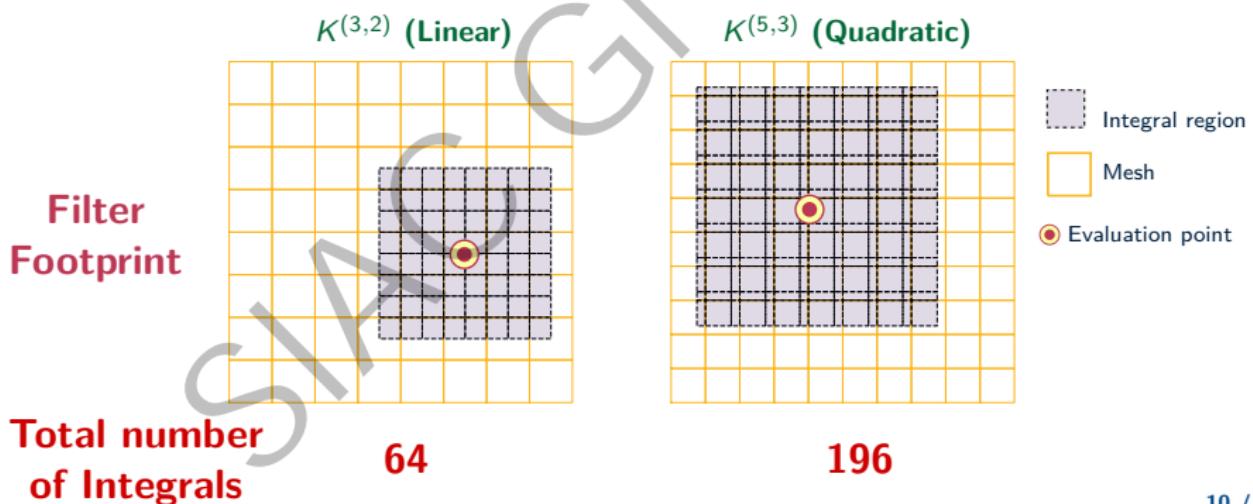
Traditional SIAC filter: tensor product structure



2D data: $K = k_x \otimes k_y$

$$data^*(\bar{x}, \bar{y}) = \frac{1}{h_x \cdot h_y} \int_{\mathbb{R}} \int_{\mathbb{R}} k_x(\bar{x} - x) \cdot k_y(\bar{y} - y) data(x, y) dy dx$$

Computation: split integral based on elements and spline breaks



LSIAC filter: a computationally efficient kernel

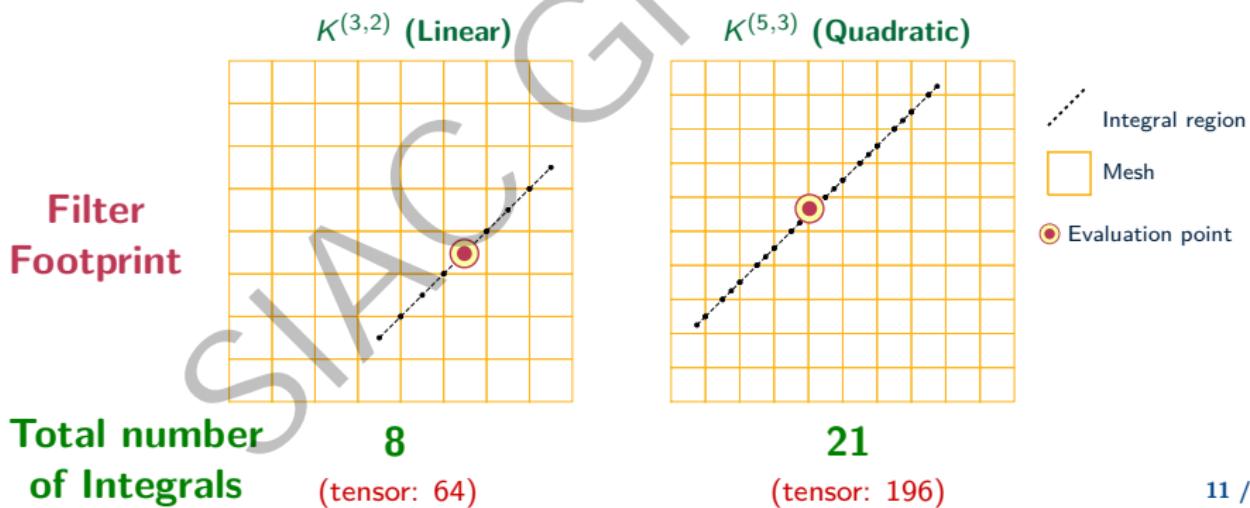
Docampo, Mirzargar, Ryan, Kirby (SISC 2017)

2D case: $K = k_\Gamma$,

$$data^*(\bar{x}, \bar{y}) = \frac{1}{h} \int_{\mathbb{R}} k_\Gamma(t) data(\Gamma(t)) dt, \quad \Gamma = (\bar{x}, \bar{y}) + h(\cos \theta, \sin \theta)$$

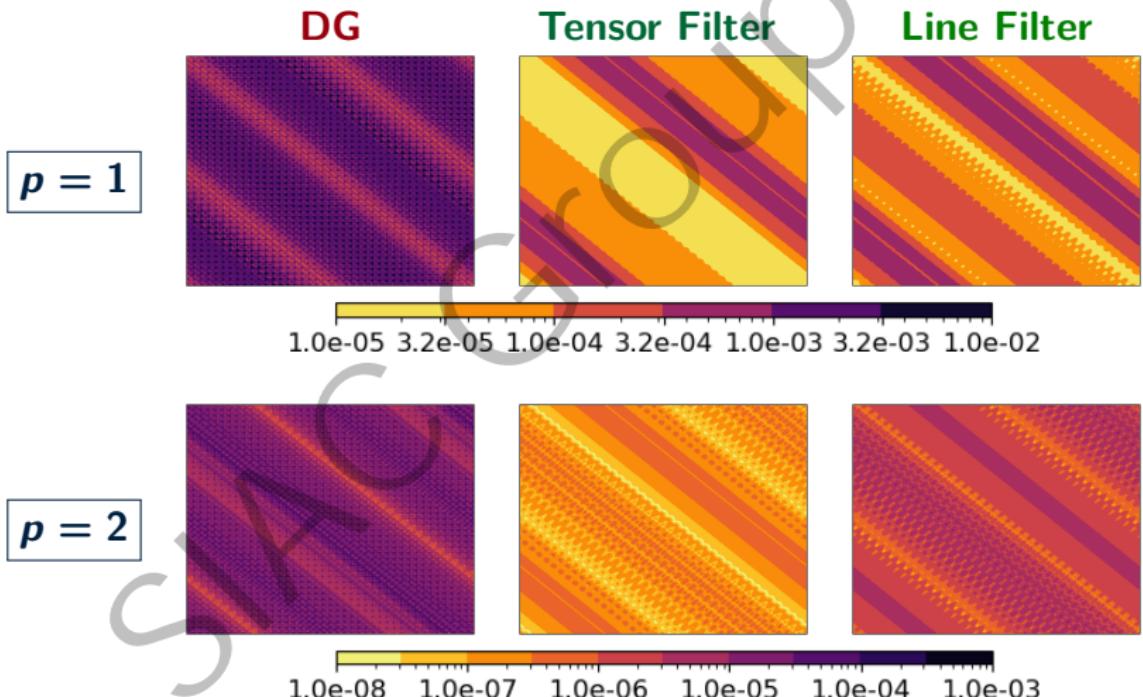


Computation: split integral based on elements and spline breaks

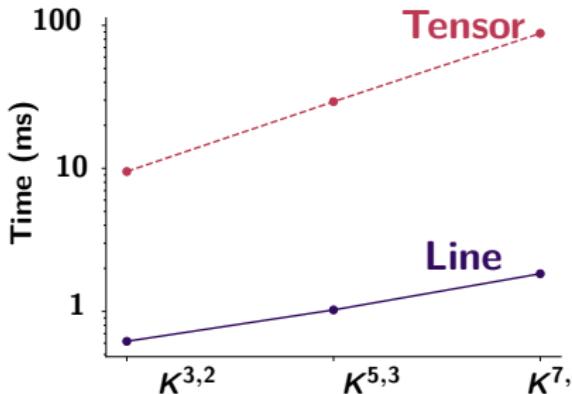


Filters performance: errors for a non-linear equation

Error contours for inviscid Burgers equation with exact solution
 $u(x, y, t = 1.0) = \sin(x + y - t)$ using 40^2 elements:



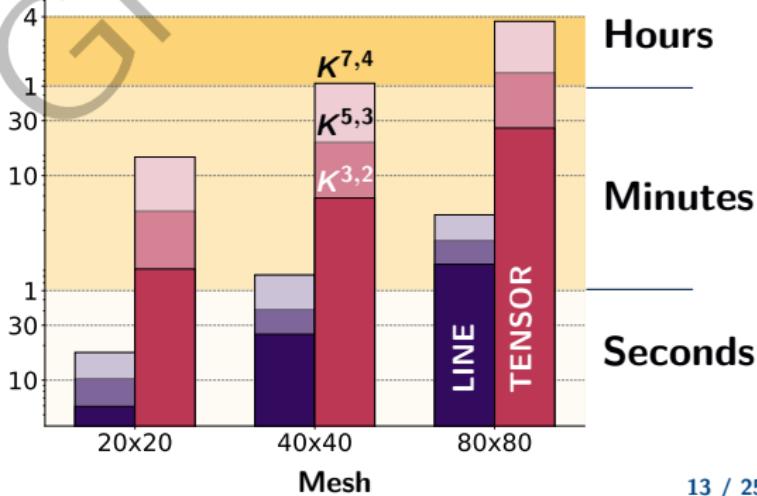
Filter performance: CPU times



Single point

Entire field

25 points per element

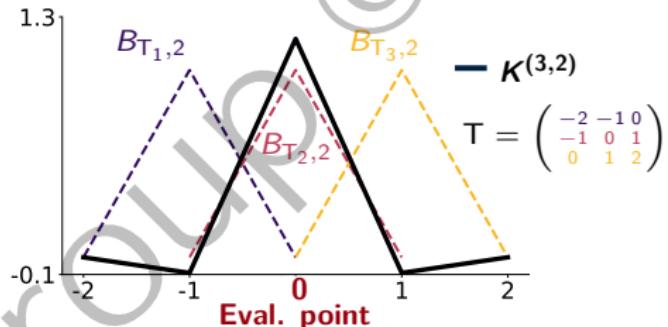


The non-symmetric RKLV (SIAC) kernel

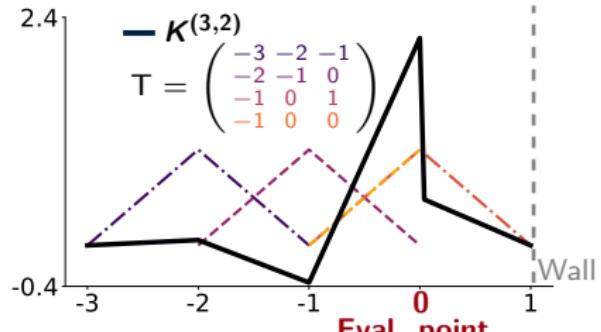
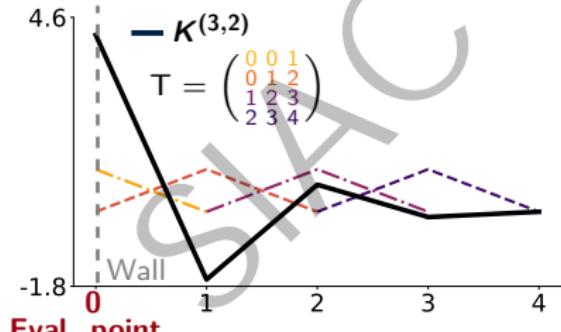
Ryan, Li, Kirby, Vuik (SISC 2014)

Symmetric filter: equal amount of information from both sides.

What if the filter doesn't fit?

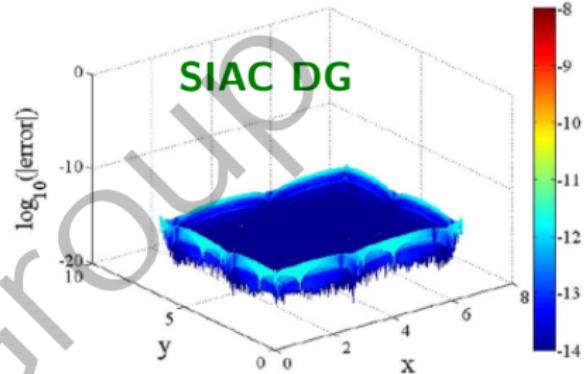
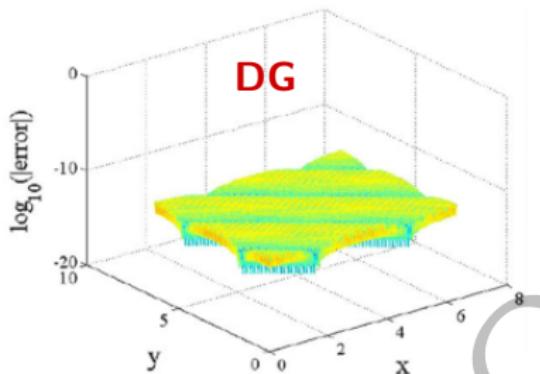


One-sided filters: shift the knot matrix



RKLV kernel: filtering near boundaries

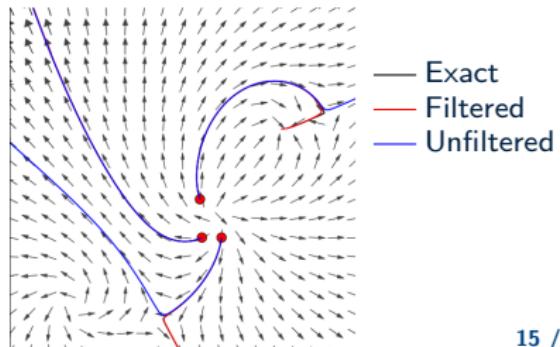
Less effective than the symmetric kernel. Yet, increased order and error reduction.



2D advection for $p = 4$ and 80×80 elements, Ryan, Li, Kirby, Vuik, (SISC 2014).

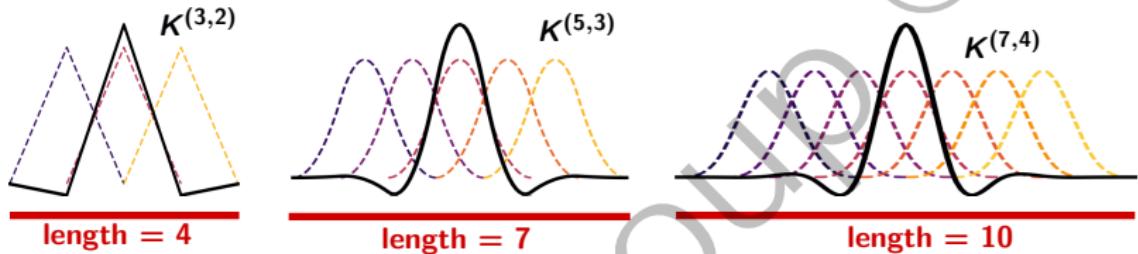
Flow visualization:
filtering (backwards) along streamlines

Docampo (PhD thesis 2017)

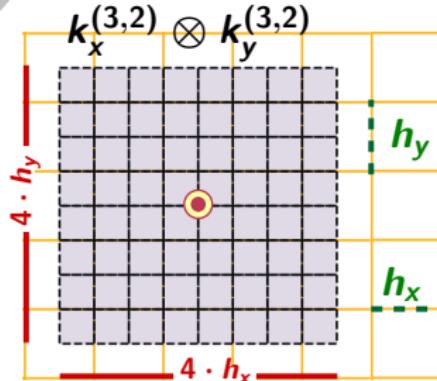
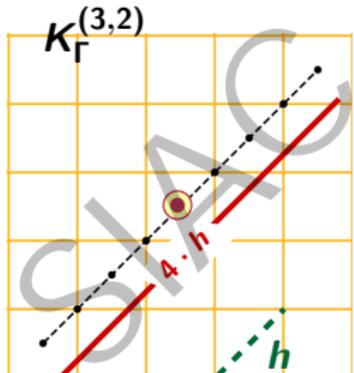


Footprint: kernel structure and scaling

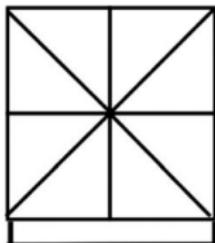
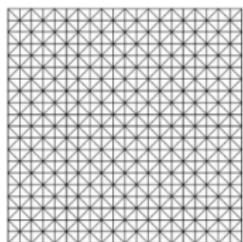
Kernel size: determined by the number & order of the B-splines



Theoretical scaling (h) for uniform meshes: element size

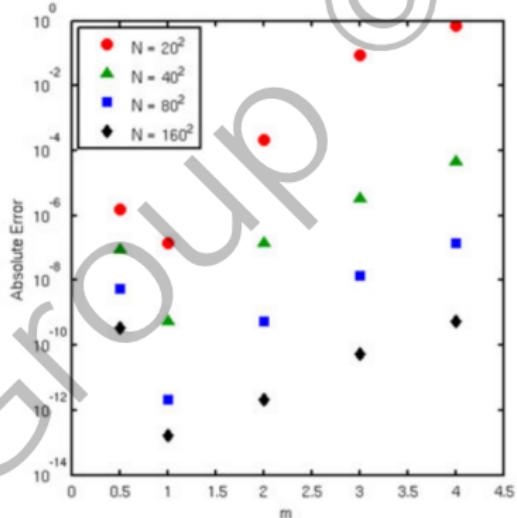


Filter performance linked to the underlying geometry



$$\text{Scaling: } h = m \cdot H$$

King, Mirzaee, Ryan, Kirby (SISC 2012)



For non-uniform meshes remains an open problem:

- Optimal scaling only found numerically
- In practise: use maximum / local element size

Post-processing data with SIAC filters

✓ Overview of the SIAC kernel

Moment preservation + B-splines + L^2 -initialization
⇒ provable $2p + 1$ superconvergence

✓ Filtering challenges

Higher dimension: tensor (accuracy) vs. line (CPU efficiency)
Domain boundaries: shifted kernels (accuracy loss)
Filter scaling: non-trivial for non-uniform meshes

➤ The software package

Code structure, geometry and tool capabilities

A standalone tool written in julia



```
function filter_data (mesh, data, parameters)
    modes = l2_projection(data)
    kernel = set_kernel(parameters)
    for point in data
        map = find_kernel_breaks(mesh, point, kernel)      # Footprint
        point* = sum( gauss(map, kernel, modes))           # Convolution
    end
end
```

- Need data file sampled at quadrature points to recover the modal form (L^2 -projection)
- **Most CPU time spent finding the filter footprint**

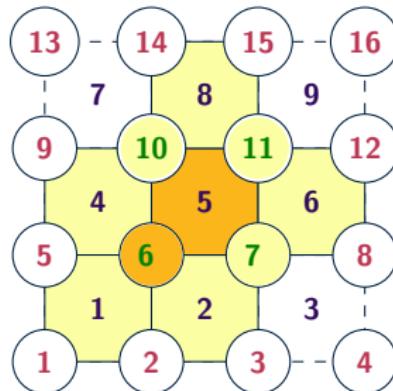
Handling the geometry: the mesh data structure

```
struct element
    nodes :: List of ordered vertex indexes forming element
    neigh :: List element neighbors
    type   :: 4 (quad) 3 (triangle)
end

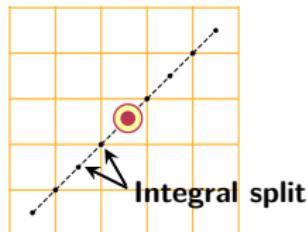
struct node
    xyz   :: vertex coordinates
    ele   :: List of elements surrounding vertex
    type  :: 1 (interior) 0 (boundary edge) -1 (boundary corner)
end
```

```
elmt[5].nodes = [6,7,11,10]
elmt[5].neigh = [2,6,8,4]
elmt[5].type  = 4
```

```
node[6].xyz  = [0.2,0.2,0.0]
node[6].ele  = [1,2,5,4]
node[6].type = 1
```



Task: collect all spline breaks and element interfaces



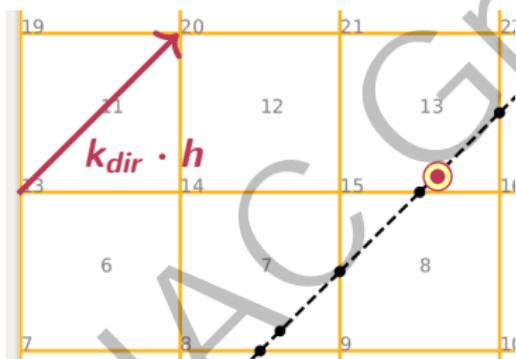
LSIAC example

$$K^{(3,2)}$$

Sorted knot matrix

$$\mathbf{T} = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

$$\mathbf{T}_- = (-2, -1)$$
$$\mathbf{T}_+ = (0, 1, 2)$$



```
for t ∈ (T_-, T_+) do
    xp = eval point, e = 13
    for i ∈ t do
        xn = kdir · h · i
        Spn = segment(xp, xn)
        if xn ∈ elmt[e] then
            store(xn, elmt[e])
        xp = xn
        else
            (j, xp) = intersect(Spn, elmt[13])
            store(xp, e, elmt[e].neigh[j])
            e = elmt[e].neigh[j] #8
        end if
    end for
end for
```

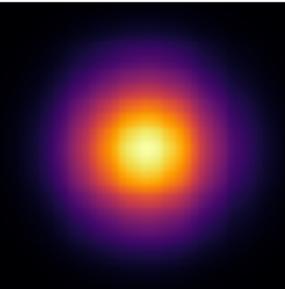
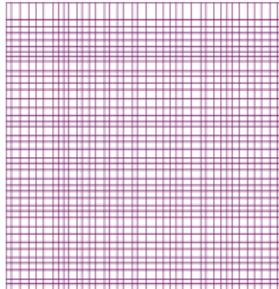
Avoids querying point location (computationally expensive)

Examples on non-uniform meshes: quads

$$p = 1$$

Mesh: 40×40

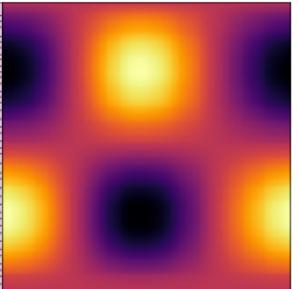
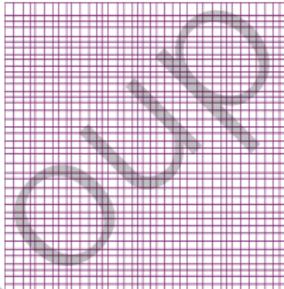
$$u = e^{-x^2-y^2}$$



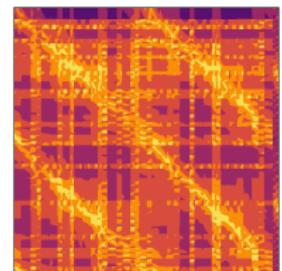
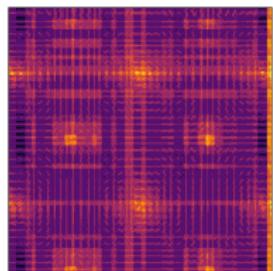
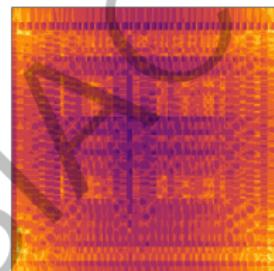
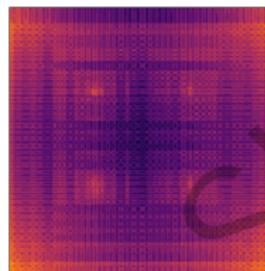
$$p = 2$$

Mesh: 40×40

$$u = \sin(x) \cos(y)$$



Error contours (log)



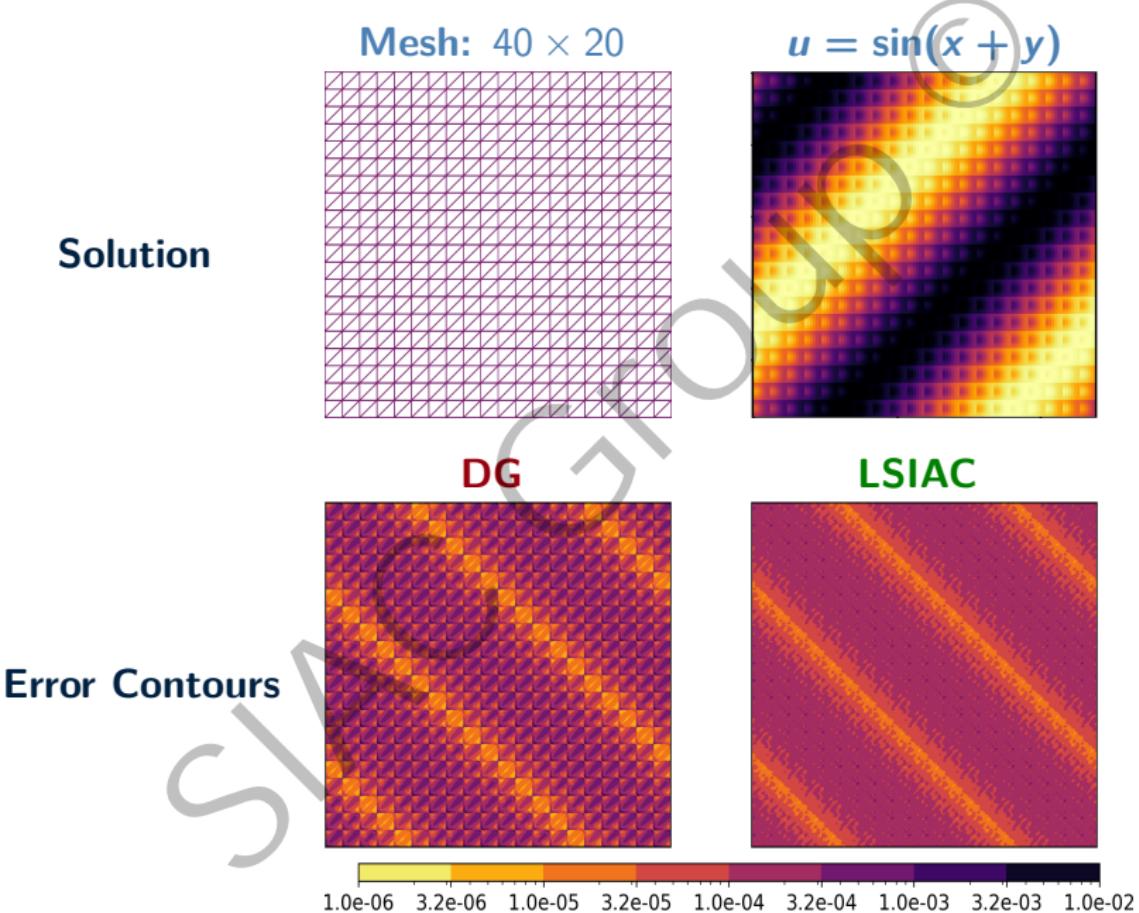
L^2 -projection

Tensor Filter

L^2 -projection

Line Filter

LSIAC filtering: triangular meshes (Matt Picklo)



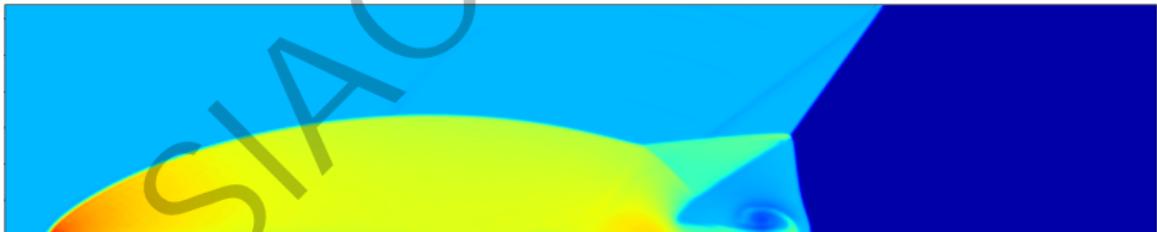
Wrapping up: hopefully you will remember...

SIAC filtering: increased smoothness and reduced errors

- Optimal superconvergent rate: $2p + 1$
- Theory: equations (linearity), mesh type & domain boundaries

Challenging implementation. Our tool currently supports:

	Kernel		Uniform		Non-uniform	
Filter	Symmetric	Shifted	Quads	Triangles	Quads	Triangles
Tensor						
Line						
		Theory & implemented		Implemented but no theory		Testing



Post-processing the DG solution ($p = 2$) to the double Mach reflection problem. Raw data: Théa Vuik and Soraya Terrab.

Grazas !!

Acknowledgements

Contributors to this talk: Matthew Picklo , Jennifer K. Ryan, Soraya Terrab and Abel Gargallo.

This filtering project is a joint initiative with Jennifer K. Ryan.

This project has received funding from The European Union's Horizon 2020 research and innovation programme under the **Marie Skłodowska-Curie grant agreement No 893378**.

