Accuracy Enhancement and Filtering for Visualisation of Discontinuous Solutions

Prof. Kees Vuik

Dr. Jennifer K. Ryan

Paulien van Slingerland

Delft University of Technology

TU Berlin, 15 December 2009

Motivation and Background

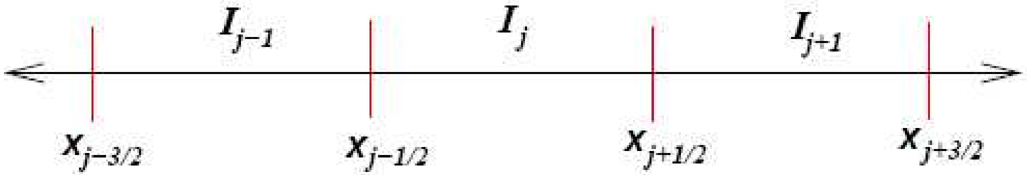
- Discontinuous Galerkin Method
- Post-Processing for Accuracy Enhancement
- Applications in Visualisation

Issues and challenges

- non-uniform mesh
- derivative post-processing
- $\Rightarrow$ one-sided post-processing $\Leftarrow$

Summary

1D Discontinuous Galerkin Formulation



Define a Mesh and an Approximation Space:

$I_j = (x_j - \frac{\triangle x_j}{2}, x_j + \frac{\triangle x_j}{2}), \quad j = 1, \cdots, N$ and $V_h = \{\phi_j^{(l)}(x) \in \mathbb{P}^k|_{I_j}, \ j = 1, \cdots, N\}$

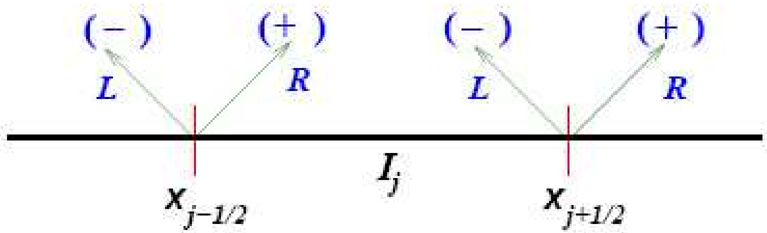
Consider $u_t + f(u)_x = 0.$

Weak Formulation: Find $u_h(x,t) \in V_h$ such that

$$\int_{I_j} (u_h)_t v dx = \int_{I_j} f(u_h) v_x dx - f((u_h)_{j+\frac{1}{2}}) v_{j+\frac{1}{2}} + f((u_h)_{j-\frac{1}{2}}) v_{j-\frac{1}{2}}$$

for all $v \in V_h$.

1D Discontinuous Galerkin Formulation



Numerical Scheme:

$$\int_{I_j} (u_h)_t v dx = \int_{I_j} f(u_h) v_x dx - \hat{f}_{j+1/2} v^-_{j+1/2} + \hat{f}_{j-1/2} v^+_{j-1/2}$$

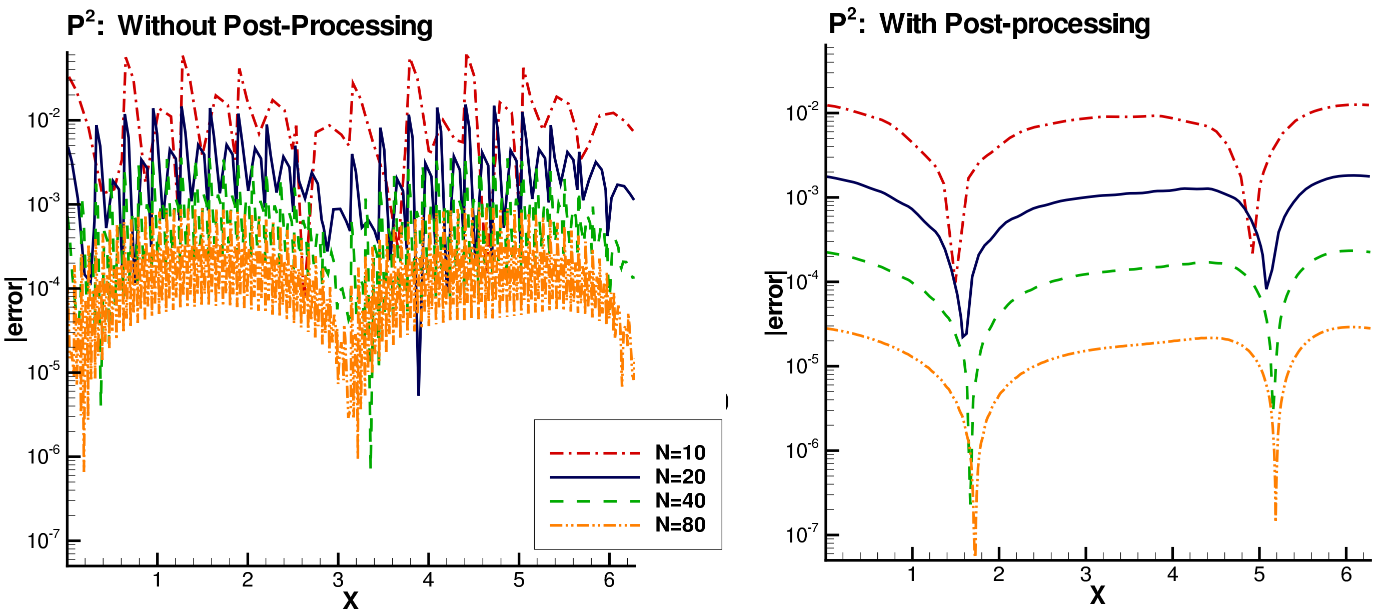
$\forall v \in V_h$.

- Use upwind monotone flux
- Take v from inside the cell

DG solution: $u_h(x,t) = \sum_{l=0}^{k} u_i^{(l)}(t)\phi_i^{(l)}(x)$ if $x \in I_i$.

Can we improve an existing DG approximation?

1-D Variable Coefficient



$u_t+(a(x)u)_x=f(x,t)$
$a(x)=2+\sin(x)$
$u(x,0)=\sin(3x)$
x in $(0,2\pi)$, $T=12.5$

Post-Processing to Improve and Approximation

The post-processor:

$$u^{\star} = K_h^{2(k+1),k+1} \star u_h$$

Why do we post-process?

- Errors in DG solution are highly oscillatory
- Post-processing filters out oscillations around the exact solution
- Result is a solution that has increased smoothness and accuracy

Post-Processor

`B. Cockburn, M. Luskin, C.-W. Shu, A. Süli, Math Comp. (2003)`

- Discontinuous Galerkin approximation errors:

$$||u_h - u||_{-l} = \mathcal{O}(h^{2k+1}),$$

whereas in the L_2-norm we have

$$||u_h - u||_2 = \mathcal{O}(h^{k+1}).$$

- Post-processor extracts this information.

$$u^*(x) = K_h * u_h$$

- Works for a locally uniform mesh:
  $\longrightarrow$ Translation invariant
  $\longrightarrow$ Post-Processor is local

Negative Order Sobolev Norm

The negative order norm is given by

$$||u||_{-\ell,\Omega} = \sup_{\phi \in \mathcal{C}_0^\infty} \frac{\int_\Omega u(x)\phi(x)dx}{||\phi||_{\ell,\Omega}}, \quad \ell \geq 1,$$

which is just a seminorm divided by the usual Sobolev norm.

Example: For the function $u_N = \sin(2\pi N x), \quad \Omega = (-1,1), \quad \ell \geq 1$, the negative order norm is

$$||u_N||_{-\ell,\Omega} = \frac{1}{(2\pi N)^\ell}$$

The negative order norm tells us that $\sin(2\pi N x)$ oscillates around zero fairly regularly.

```
Bramble & Schatz, Math.  Comp.  (1977)
Mock & Lax, Comm.  Pure Appl.  Math (1978)
```

Post-Processor Kernel

- Independent of the partial differential equation.
- Applied only at the final time.
- Filters out oscillations in the error.

Kernel Properties

- Compact Support $\Rightarrow$ Computationally advantages
- Reproduces polynomials of degree $2k$ by convolution. $\Rightarrow$ Accuracy is not lost.
- Linear combination of B-splines.

Post-Processor

- Use Negative order norms $\Rightarrow$ Tells us how oscillatory a function is (difficult to compute).
- Use Convolution $\Rightarrow$ “Filters” out these oscillations
- B-splines $\Rightarrow$ Gives the convolution kernel nice properties.
- Make assumptions on the approximation and the mesh.

`Result:` A post-processor that filters out oscillations in the error and improves the order of accuracy.

Kernel Construction

Post-processed solution: $u^\star(x) = K_h^{2(k+1),k+1} \star u_h$.

$$K_h^{2(k+1),k+1}(x) = \frac{1}{h}\sum_{\gamma=-k}^{k} c_\gamma^{2(k+1),k+1}\psi^{(k+1)}\left(\frac{x}{h}-\gamma\right)$$

$h = \triangle x_i$ for all i, and $c_\gamma^{2(k+1),k+1} \in \mathbb{R}$.

B-spline recursion formula:

$$\psi^{(1)} = \chi_{[-1/2,1/2]},$$

$$\psi^{(k+1)} = \frac{1}{k}\left[\left(x+\frac{k+1}{2}\right)\psi^{(k)}\left(x+\frac{1}{2}\right)+\left(\frac{k+1}{2}-x\right)\psi^{(k)}\left(x-\frac{1}{2}\right)\right], \quad k \geq 1.$$

Convolution Coefficients

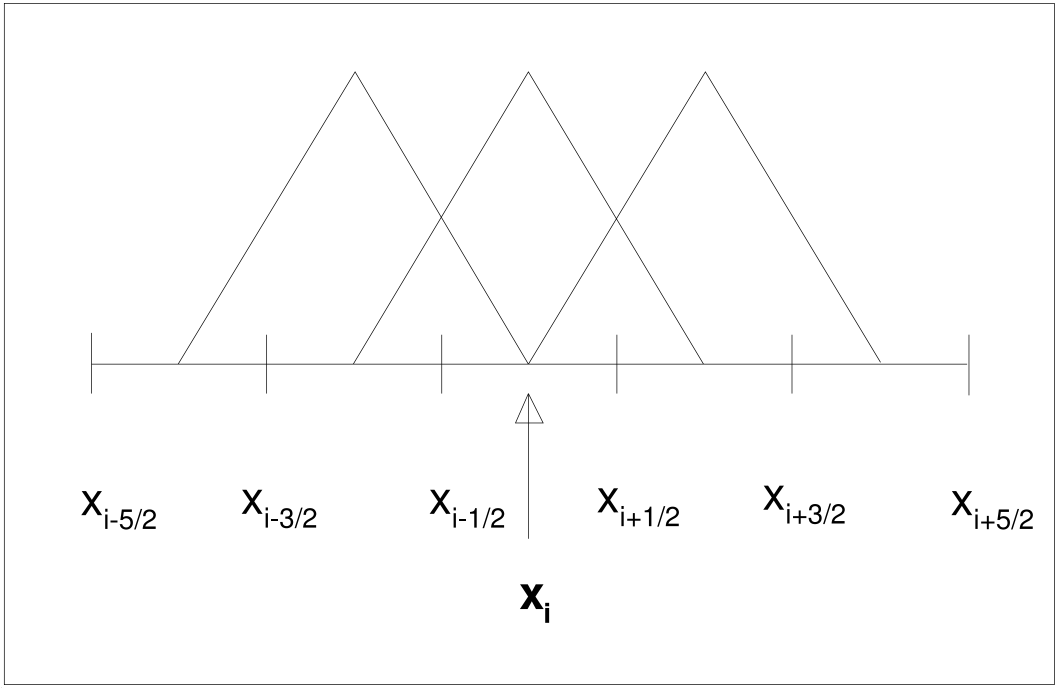
To find $c_\gamma,\ \gamma = -k, \cdots, k:$
Use $K_h^{2(k+1),k+1} \star x^m = x^m$ for $m = 1, \cdots, x^{2k}$

$$\begin{bmatrix} \int \psi^{(k+1)}(x-y-k)\,dy & \cdots & \int \psi^{(k+1)}(x-y+k)\,dy \\ \int \psi^{(k+1)}(x-y-k)y\,dy & \cdots & \int \psi^{(k+1)}(x-y+k)y\,dy \\ \int \psi^{(k+1)}(x-y-k)y^2\,dy & \cdots & \int \psi^{(k+1)}(x-y+k)y^2\,dy \\ \cdots & \cdots & \cdots \\ \int \psi^{(k+1)}(x-y-k)y^{2k}\,dy & \cdots & \int \psi^{(k+1)}(x-y+k)y^{2k}\,dy \end{bmatrix} \begin{bmatrix} c_{-k} \\ \cdots \\ c_0 \\ \cdots \\ c_k \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \cdots & x^{k+1} & \cdots & x^{2k} \end{bmatrix}^T$$

Example: Kernel B-splines

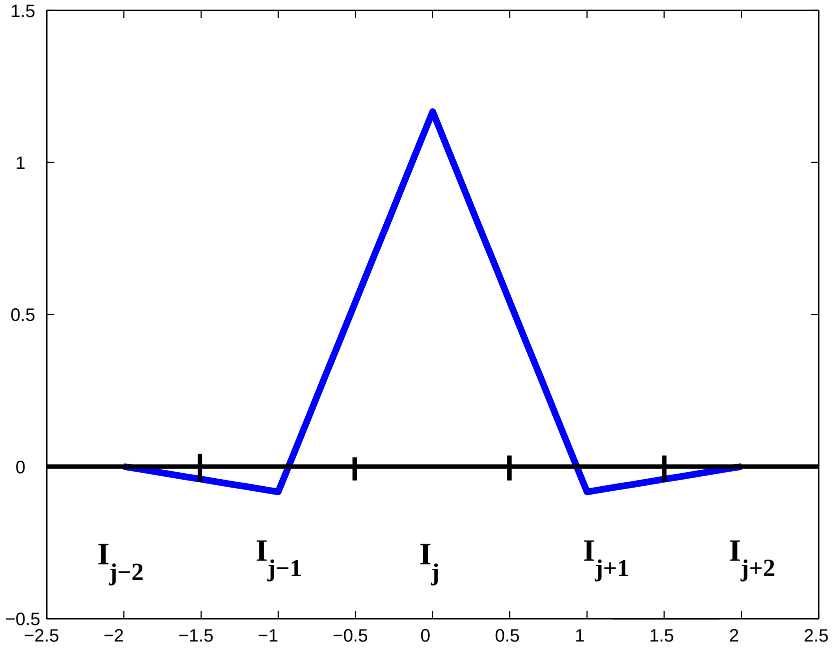
Second Order Approximation



$$\psi^{(2)}(x+1) \qquad \psi^{(2)}(x) \qquad \psi^{(2)}(x-1)$$

Kernel for Linear Approximation

Find $c_\gamma,\ \gamma = -1, 0, 1$: Use $K_h^{4,2} \star p = p$ for $p = 1, x, x^2$



$$K^{4,2}(x) = \frac{-1}{12}\psi^{(2)}(x-1) + \frac{7}{6}\psi^{(2)}(x) - \frac{1}{12}\psi^{(2)}(x+1)$$

Implementing the Post-processor

For element $I_j = (x_{j-1/2,j+1/2})$:

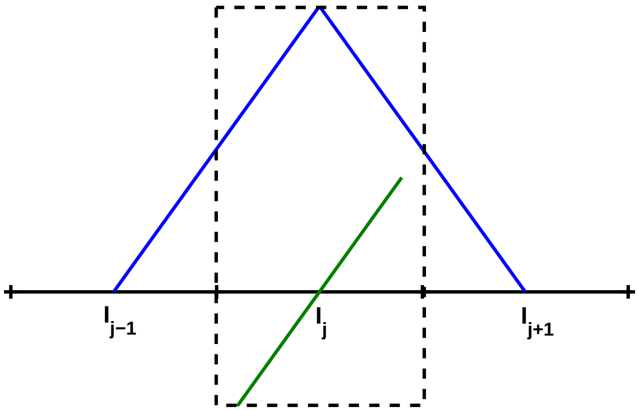
$$\Rightarrow u^\star(x) = \sum_i \sum_{l=0}^{k} u_i^l \sum_{\gamma=-k}^{k} c_\gamma^{2(k+1),k+1} \int \psi^{(k+1)}\left(\frac{x-y}{h} - \gamma\right) \phi_i^{(l)}(y)\, dy.$$

where $i = j - p', \cdots, j + p',\ p' = \lceil \frac{3k+1}{2} \rceil$

k	1	2	3
p'	2	3	5

Note: p' is the number of elements needed on each side of the element being post-processed.

Example: Implementing $k = 1$ case



green line = DG approximation on one element.

blue line = kernel. The kernel is introducing smoothness at the element boundaries.

Convolution Kernel:

$$K^{4,2}(x) = \frac{-1}{12}\psi^{(2)}(x-1) + \frac{7}{6}\psi^{(2)}(x) - \frac{1}{12}\psi^{(2)}(x+1)$$

Discontinuous Galerkin Solution: $u_h(x) = u_j^{(0)}\phi_j^{(0)} + u_j^{(1)}\phi_j^{(1)}$ on element $I_j = (x_{j-1/2}, x_{j+1/2})$.

$2-D$ *Kernel*

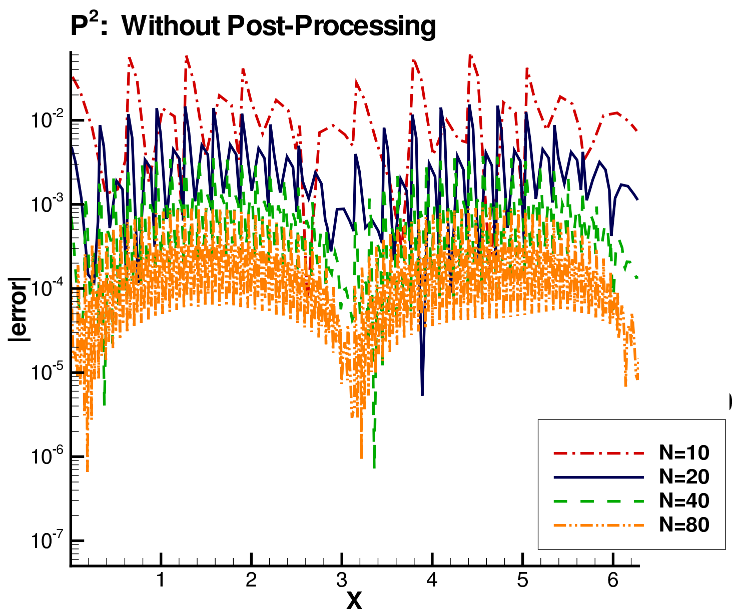
The $2-$D case is simply a tensor product of the $1-$D case.

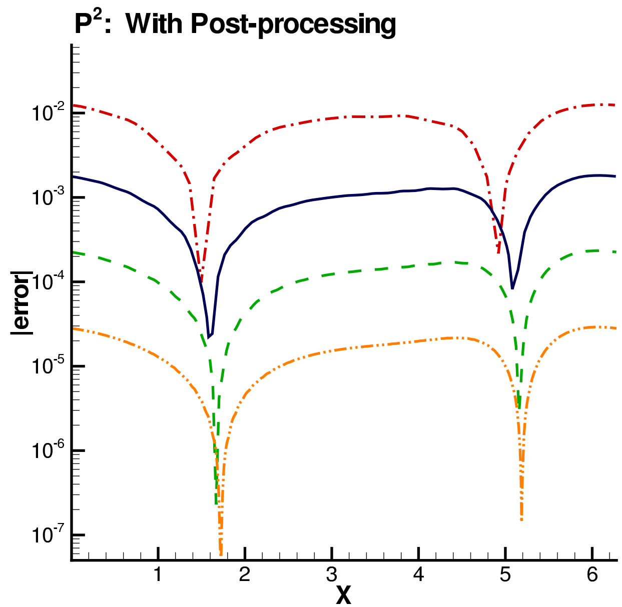
Kernel:

$$K_h = \frac{1}{h_x h_y} \sum_{\gamma_x=-k}^{k} \sum_{\gamma_y=-k}^{k} c_{\gamma_x} c_{\gamma_y} \psi^{(k+1)}\left(\frac{x}{h_x} - \gamma_x\right) \psi^{(k+1)}\left(\frac{y}{h_y} - \gamma_y\right)$$

We can use *either* a tensor product of polynomials, $\mathbb{Q}^k$ - ($\{1, x, y, xy\}$), or the usual polynomial basis, $\mathbb{P}^k$ - ($\{1, x, y\}$).

1-D Variable Coefficient





$$u_t+(a(x)u)_x=f(x,t)$$
$$a(x)=2+\sin(x)$$
$$u(x,0)=\sin(3x)$$
$$x \text{ in } (0,2pi),\ T=12.5$$

$1-D$ *Variable Coefficient Equation*

Ryan, Shu, Atkins, SISC (2005)

	$u_h(x, 12.5)$		$u^*(x, 12.5)$	
mesh	L^2 error	order	L^2 error	order
		$\mathbb{P}^1$		
10	1.83E-02	—	7.82E-02	—
20	4.35E-03	2.07	1.08E-03	2.86
40	1.07E-03	2.03	1.39E-04	2.96
		$\mathbb{P}^2$		
10	8.61E-04	—	1.34E-04	—
20	1.07E-04	3.01	2.34E-06	5.84
40	1.34E-05	3.00	4.55E-08	5.69

$$u_t + (au)_x = f$$

$$\begin{aligned} a(x) &= 2 + \sin(x) \\ u(x,0) &= \sin(3x) \\ u(0,t) &= u(2\pi, t) \\ T &= 12.5 \end{aligned}$$

TUDelft

Streamline Calculation: Filtering Entire Field

- Obtain numerical approximation
- Post-Process the approximation
- We can then choose our time integrator for the streamline calculation (such as RK-4)

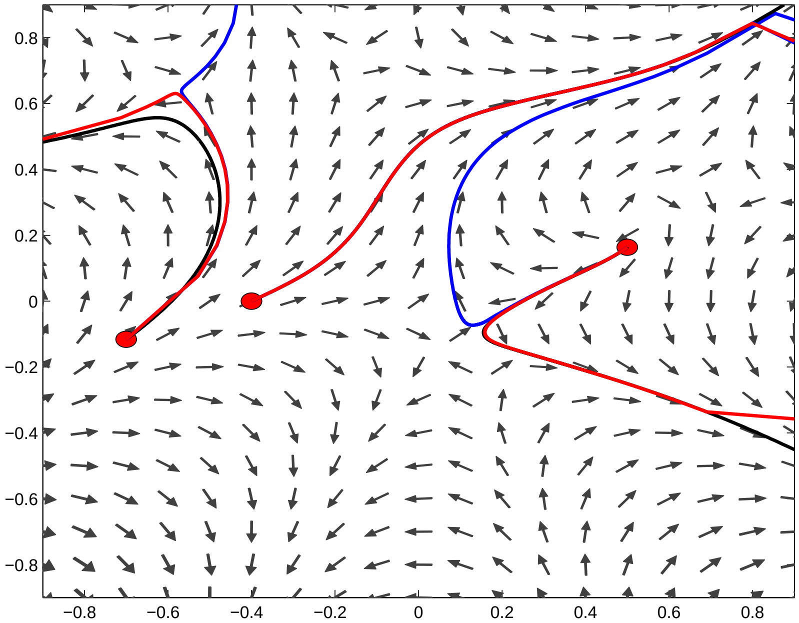
$$\begin{aligned} \frac{d}{dt}\vec{x}(t) &= \vec{F}(\vec{x}(t)) \\ \vec{x}(t=0) &= \vec{x}_0 \end{aligned}$$

- The post-processor increases smoothness of the approximation to help obtain the correct streamline.

Applications in Filtering for Streamline Visualisation

Example Field: Scheuerman, Tricoche, and Hagen, IEEE Vis (1999).

Steffan, Curtis, Kirby, and Ryan, IEEE-TVCG (2008).



$$
\begin{aligned}
z &= x + \imath y \\
u &= Re(r) \\
v &= -Im(r).
\end{aligned}
$$

$$
\begin{aligned}
r = \; & (z - (0.74 + 0.35\imath))(z - (0.68 - 0.59\imath) \\
& (z - (-0.11 - 0.72\imath))(\bar{z} - (-0.58 + 0. \\
& (\bar{z} - (0.51 - 0.27\imath))(\bar{z} - (-0.12 + 0.84
\end{aligned}
$$

$$(u, v)^T = \vec{F}(x, y), \quad \Omega = [-1, 1] \times [-1, 1]$$

Streamline Calculation: Filtering Entire Field

U component

	L^2 error		L^∞ error	
N	BEFORE	AFTER	BEFORE	AFTER
	$\mathbb{P}^1$			
20	1.2642E-02	4.8779E-04	1.3028E-01	2.0830E-03
40	4.4291E-03	3.8597E-05	4.8341E-02	1.7929E-04
80	1.3054E-03	2.7114E-06	1.7165E-02	1.3033E-05
	$\mathbb{P}^2$			
20	2.2576E-04	6.8329E-06	1.8986E-03	1.3061E-05
40	5.0880E-05	1.4086E-07	5.4698E-04	2.6435E-07
80	8.4056E-06	2.4689E-09	9.9905E-05	4.6007E-09

Applications in Filtering for Visualization

Streamline Calculation: Filtering Entire Field

Limitations:

- Uniform quadrilateral mesh $\cdots$ What about $3-D$?
  $\Rightarrow$ For 1 & 2-D use a characteristic length.$\Leftarrow$
- Higher order streamline integrator - need derivative information.
  $\longrightarrow$ Use smoother splines.
- Maintaining Boundary Values.
- Post-Processing entire field can be expensive (R.M. Kirby, Utah).

Nonuniform Mesh: Characteristic Length

Curtis, Kirby, Ryan, and Shu, SISC (2007).

Post-processing solution on cell I_j.

- Let L be the characteristic length used in the post-processor, where $L = \max_{i=1,\cdots,N} \triangle x_i$.

$$C_L(i,l,k,x) = \frac{1}{L}\int_{I_{i+j}} \psi^{(k+1)}\left(\frac{y-x}{L} - \gamma\right)\left(\frac{y - x_{i+j}}{\triangle x_{i+j}}\right)^l dy,$$

- Find post-processed solution on I_j :

$$u^\star(x) = \sum_{i=-p'}^{p'} \sum_{l=0}^{k} u_{(i+j)}^{(l)} C_L(i,l,k,x)$$

Streamline Calculation: Filtering Entire Field

Limitations:

- Uniform quadrilateral mesh $\cdots$ What about $3-D$?
  $\longrightarrow$ For 1 & 2-D use a characteristic length.
- $\Rightarrow$Higher order streamline integrator - need derivative information.$\Leftarrow$
  $\longrightarrow$ Use smoother splines.
- Maintaining Boundary Values.
- Post-Processing entire field can be expensive (R.M. Kirby, Utah).

Accuracy Improvement for Derivatives

Two methods

- Calculating the derivative of the post-processing polynomial directly.

  `Ryan, Shu, Atkins, SISC (2005)`

$$\Rightarrow \mathcal{O}(h^{2k+2-d})$$

- $\Rightarrow$ Using higher-order B-splines in the convolution kernel together with divided differences of the numerical solution. $\Leftarrow$

  `Thomee, Math. Comp. (1977)`

  `Cockburn & Ryan, JCP (2009)`

$$\Rightarrow \mathcal{O}(h^{2k+1})$$

Accuracy Improvement for Derivatives: *Higher Order Splines*

$$\frac{d^s u^*}{dx^s}(x) = \frac{1}{h}\int_{-\infty}^{\infty} \tilde{K}^{s,2(k+1),k+1}\left(\frac{y-x}{h}\right)\partial_h^s u_h(y,T)\,dy.$$

for the s^{th} derivative.

- Uses higher order B-splines than post-processed solution.
- Kernel has a wider support.

Kernel:

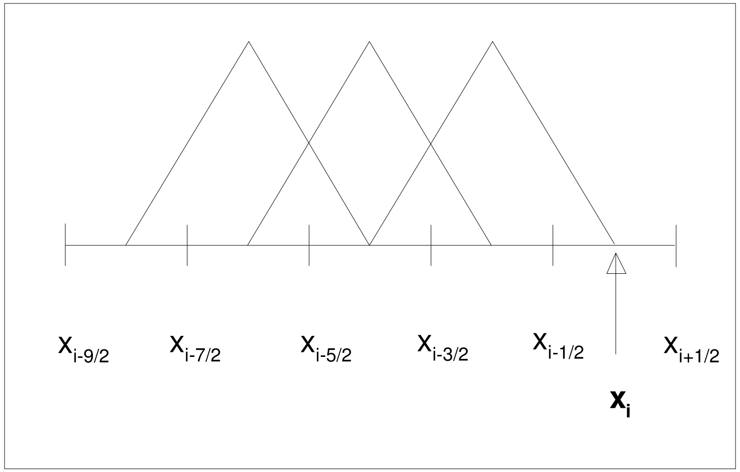
$$\tilde{K}^{s,2(k+1),k+1} = \sum_{\gamma=-k}^{k} \tilde{c}_\gamma\, \psi^{(k+s+1)}(x-\gamma).$$

Streamline Calculation: Filtering Entire Field

Limitations:

- Uniform quadralateral mesh $\cdots$ What about $3-D$?
  $\longrightarrow$ For 1 & 2-D use a characteristic length.
- Higher order streamline integrator - need derivative information.
  $\longrightarrow$ Use smoother splines.
- $\Longrightarrow$ Maintaining Boundary Values. $\Longleftarrow$
- Post-Processing entire field can be expensive (R.M. Kirby, Utah).

(Old) Left Post-Processor



Ryan and Shu, MAA (2003)

$$u^\star(x) = \sum_{j=-2p'}^{0} \sum_{l=0}^{k} u_{i+j}^{(l)}\, C(j,l,k,x)$$

where $p' = \lceil (3k+1)/2 \rceil \leq 2k$
and $u^\star \in \mathbb{P}^{2k+1}$

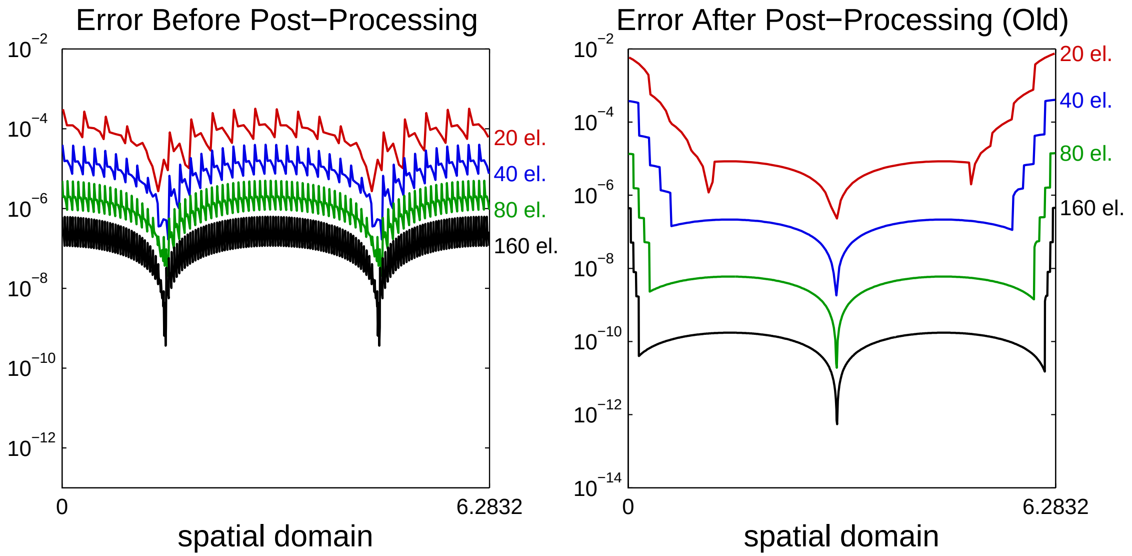
$$C(j,l,k,x) = \frac{1}{h} \sum_{\gamma=-2k-1}^{-k} c_\gamma^{2(k+1),k+1} \int_{-\frac{1}{2}-(\xi_i+\gamma)}^{\frac{1}{2}-(\xi_i+\gamma)} \psi^{(k+1)}(\eta)\,(\xi_i+\eta+\gamma-j)^l\, dy$$

For $k = 1$:

$$K(x) = \frac{11}{12}\psi^{(2)}(x+3) - \frac{17}{6}\psi^{(2)}(x+2) + \frac{35}{12}\psi^{(2)}(x+1)$$

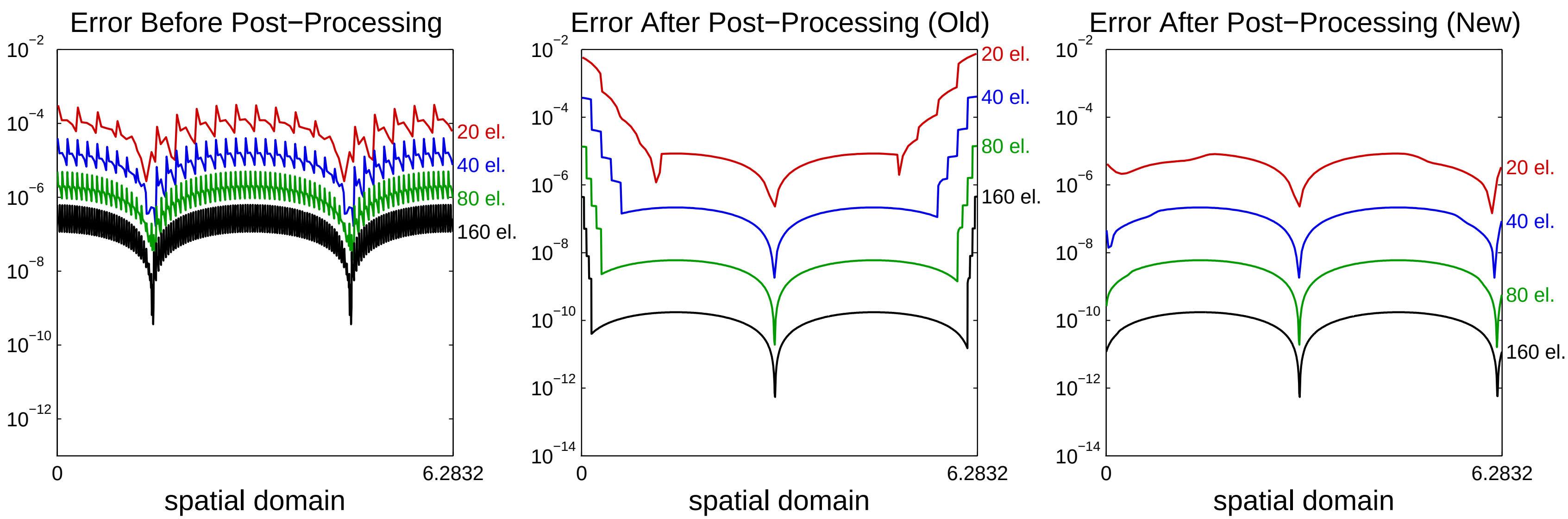
Old One-Sided Post-Processing: : $u_t + u_x = 0$, *periodic BC*

Problem 1: discontinuities are not eliminated (stair-stepping)
Problem 2: the errors at the boundary can be worse than before



New One-Sided Post-Processing: : $u_t + u_x = 0$, *periodic BC*

Problem 1: not all discontinuities are eliminated (stair-stepping)
Problem 2: the errors at the boundary can be worse than before



These problems can be solved through a new type of one-sided post-processing (following slides)

New One-Sided Post-Processing

The discontinuities can be avoided by using kernel nodes that depend continuously on the evaluation point through the shift function $\lambda(\bar{x})$:

$$u_h^\star(\bar{x}) = \sum_{\gamma=0}^{2k} c_\gamma(\bar{x}) \int_I \psi_h^{(k+1)}\Big(x - \underbrace{(\lambda(\bar{x}) + \gamma)}_{\text{kernel node}}\Big)\, u_h(\bar{x} - x)\, dx.$$

van Slingerland, Ryan, & Vuik (2009).

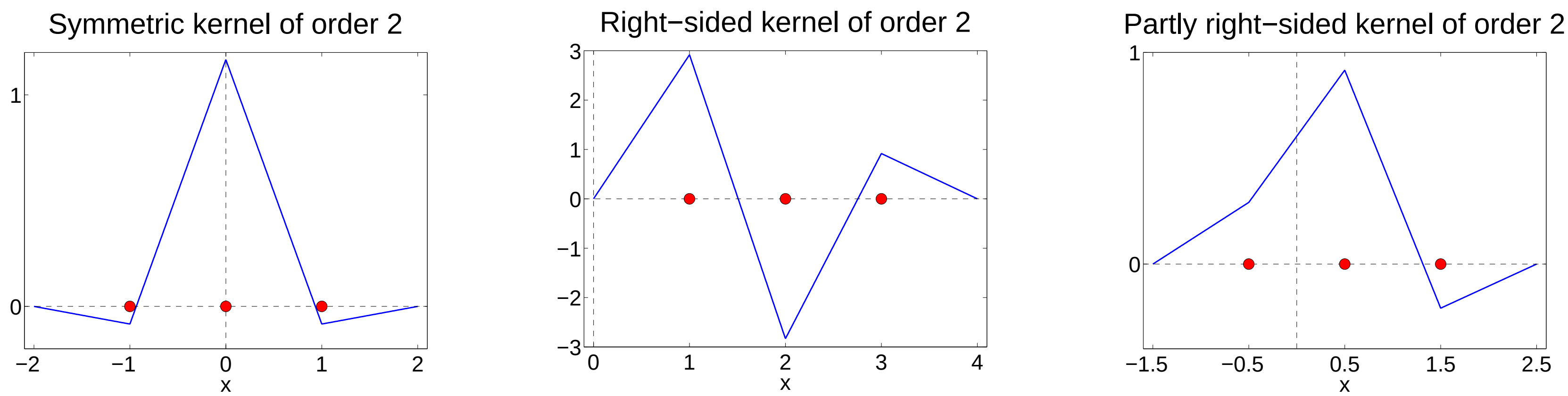
New One-Sided Post-Processing

The discontinuities can be avoided by using kernel nodes that depend continuously on the evaluation point through the shift function $\lambda(\bar{x})$:

$$u_h^\star(\bar{x}) = \sum_{\gamma=0}^{2k} c_\gamma(\bar{x}) \int_I \psi_h^{(k+1)}\Big(x - \underbrace{(\lambda(\bar{x}) + \gamma)}_{\text{kernel node}}\Big)\, u_h(\bar{x} - x)\, dx.$$

Three examples (the kernel nodes are indicated by the red circles):

$\lambda(\bar{x}) = -k$ | $\lambda(\bar{x}) = \frac{k+1}{2}$ | $\lambda(\bar{x}) = -0.5$



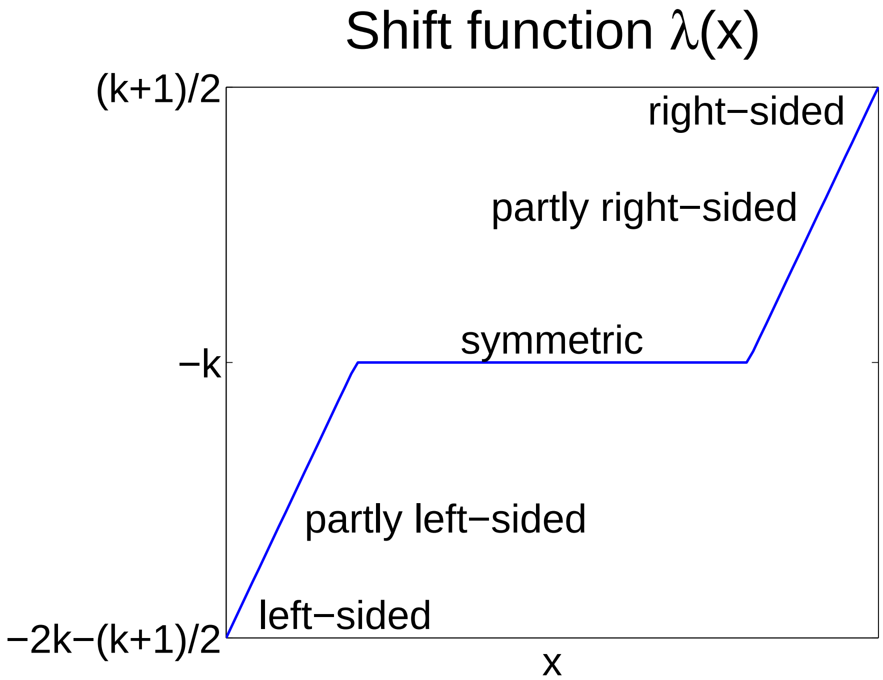
Use in the domain interior. | Use at the left boundary | Use near the left boundary

TU Delft

New One-Sided Post-Processing

The discontinuities can be avoided by using kernel nodes that depend continuously on the evaluation point through the shift function $\lambda(\bar{x})$:

$$u_h^\star(\bar{x}) = \sum_{\gamma=0}^{2k} c_\gamma(\bar{x}) \int_I \psi_h^{(k+1)}\big(x - \underbrace{(\lambda(\bar{x}) + \gamma)}_{\text{kernel node}}\big)\, u_h(\bar{x} - x)\, dx.$$



New One-Sided Post-Processing

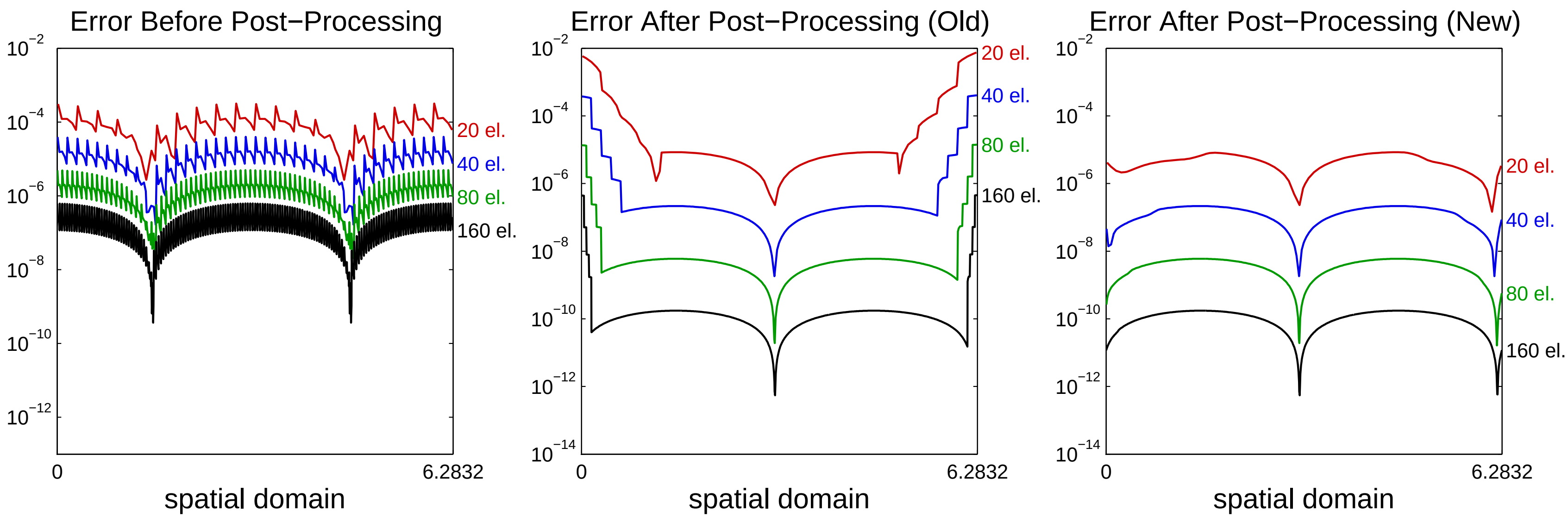
The accuracy near the boundary can be improved by using extra kernel nodes in that region.

$$\underbrace{u_h^\star(\bar{x}) = \theta(\bar{x}) \underbrace{u^\star_{h,2k+1}(\bar{x})}_{\text{filtering with } 2k+1 \text{ nodes}} + (1-\theta(\bar{x})) \underbrace{u^\star_{h,4k+1}(\bar{x})}_{\text{filtering with } 4k+1 \text{ nodes}}}_{\text{smooth convex combination}}$$

- In the interior: $\theta(\bar{x}) = 1$ (old filter suffices)
- Near the boundary: $\theta(\bar{x}) = 0$ (extra accuracy through extra nodes)
- Transition regions: choose θ smooth

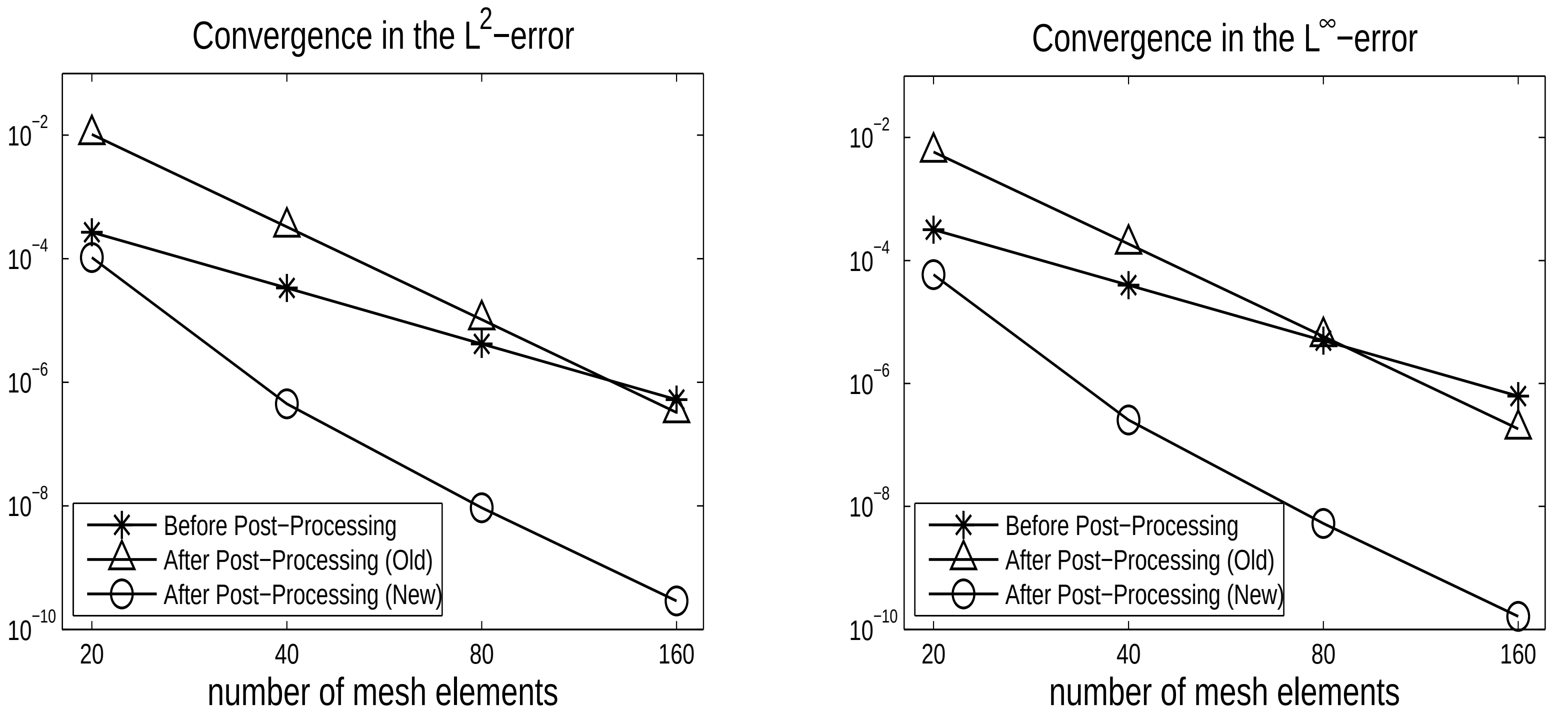
New One-Sided Post-Processing: $u_t + u_x = 0$, *periodic BC*

The new post-processor improves both the convergence rate and the absolute value of the errors for a problem with a periodic BC



New One-Sided Post-Processing: $u_t + u_x = 0$, *periodic BC*

The new post-processor improves both the convergence rate and the absolute value of the errors for a problem with a periodic BC



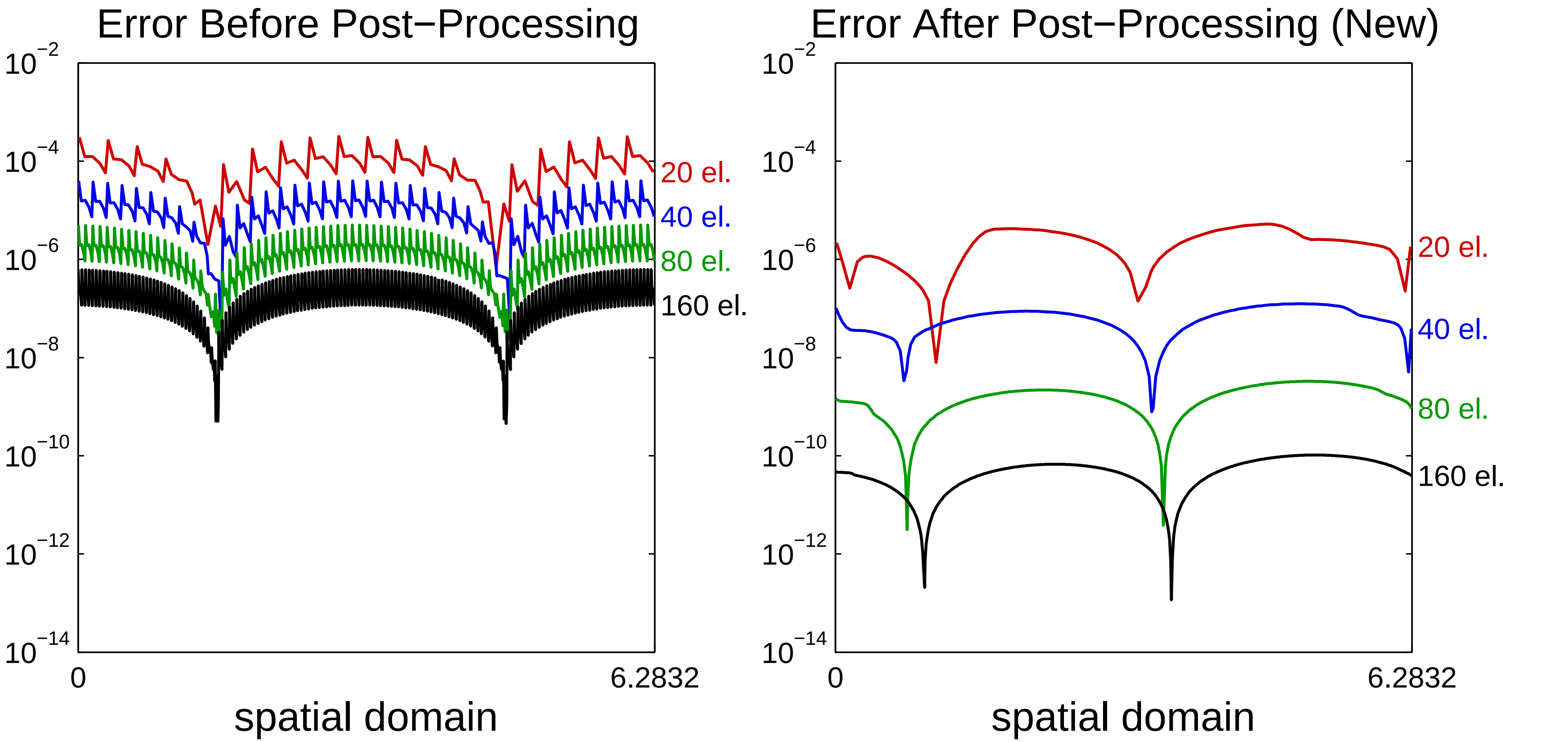
New One-Sided Post-Processing: $u_t + u_x = 0$, *periodic BC*

The new post-processor improves both the convergence rate and the absolute value of the errors for a problem with a periodic BC

	Before		**After (Old)**		**After (New)**	
mesh	L^2-error	order	L^2-error	order	L^2-error	order
	Polynomial Degree k = 2					
20	2.683e-04	-	4.003e-03	-	1.301e-05	-
40	3.352e-05	3.00	2.108e-04	4.25	3.767e-07	5.11
80	4.190e-06	3.00	5.464e-06	5.27	1.056e-08	5.16
160	5.238e-07	3.00	1.254e-07	5.45	3.090e-10	5.10
	Polynomial Degree k = 3					
20	5.176e-06	-	1.304e-04	-	3.757e-07	-
40	3.236e-07	4.00	4.712e-06	4.79	6.634e-10	9.15
80	2.023e-08	4.00	3.406e-08	7.11	2.957e-12	7.81
160	1.264e-09	4.00	1.999e-10	7.41	1.287e-14	7.84

New One-Sided Post-Processing: $u_t + u_x = 0$, *Dirichlet BC*

The new post-processor improves both the convergence rate and the absolute value of the errors for a problem with a Dirichlet BC



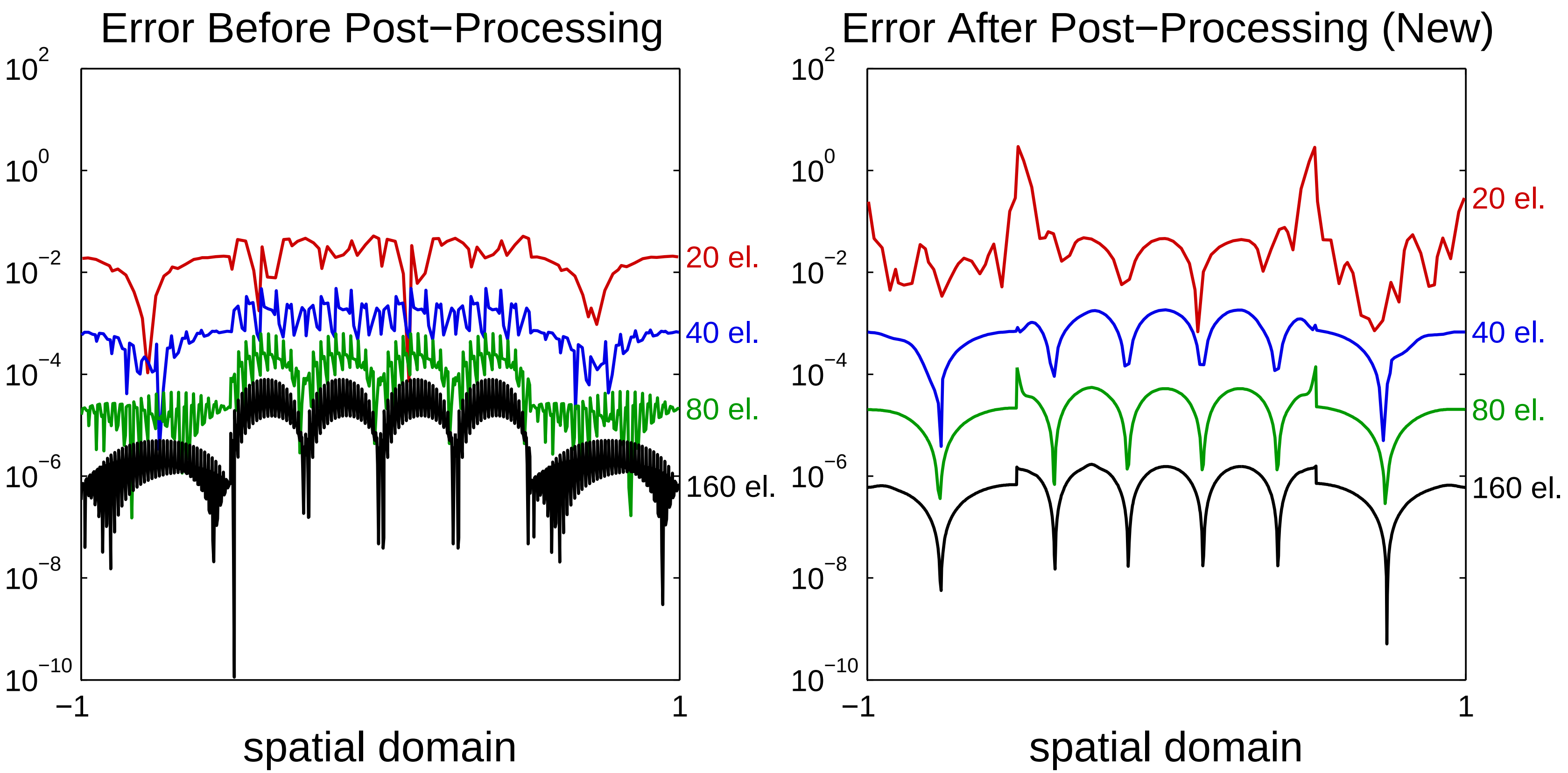
New One-Sided Post-Processing: $u_t + u_x = 0$, *Dirichlet BC*

The new post-processor improves both the convergence rate and the absolute value of the errors for a problem with a Dirichlet BC

	Before		**After (Old)**		**After (New)**	
mesh	L^2-error	order	L^2-error	order	L^2-error	order
	Polynomial Degree k = 2					
20	2.681e-04	-	4.003e-03	-	6.984e-06	-
40	3.352e-05	3.00	2.108e-04	4.25	1.850e-07	5.24
80	4.190e-06	3.00	5.464e-06	5.27	4.798e-09	5.27
160	5.238e-07	3.00	1.254e-07	5.45	1.498e-10	5.00
	Polynomial Degree k = 3					
20	5.176e-06	-	1.304e-04	-	3.751e-07	-
40	3.236e-07	4.00	4.712e-06	4.79	6.396e-10	9.20
80	2.023e-08	4.00	3.406e-08	7.11	2.867e-12	7.80
160	1.264e-09	4.00	1.999e-10	7.41	3.079e-14	6.54

New One-Sided Post-Processing: $u_t + au_x = 0$, *a discontinuous*

For this problem with two stationary shocks, the post-processor requires a sufficiently fine mesh



New One-Sided Post-Processing: $u_t + au_x = 0$, *a discontinuous*

For this problem with two stationary shocks, the post-processor requires a sufficiently fine mesh

	Before		After (Old)		After (New)	
mesh	L^2-error	order	L^2-error	order	L^2-error	order
	Polynomial Degree k = 2					
20	3.646e-02	-	6.808e+00	-	5.709e-01	-
40	2.052e-03	4.15	1.672e-01	5.35	1.249e-03	8.84
80	2.173e-04	3.24	6.027e-03	4.79	4.166e-05	4.91
160	2.682e-05	3.02	8.414e-05	6.16	1.181e-06	5.14
	Polynomial Degree k = 3					
20	1.085e-03	-	3.579e+00	-	2.270e-01	-
40	6.602e-05	4.04	1.865e-02	7.58	2.640e-03	6.43
80	4.132e-06	4.00	6.502e-04	4.84	5.205e-06	8.99
160	2.584e-07	4.00	2.623e-06	7.95	4.670e-09	10.12

Summary

- Using B-splines allows us to induce smoothness on the DG field and enhance accuracy.
- We can obtain this improvement from order k+1 to order 2k+1 for smoothly varying meshes as well as derivatives of the DG solution.
- Recent extensions allow us to have the improvement in accuracy near the boundaries as well.
  - The kernel is adjusted according to the point we would like to post-process.
  - Near the boundary, we use more kernel nodes.
- We can use this post-processing technique as a visualisation tool to maintain more accurate streamlines.

Acknowledgments: This research is supported by the U.S. Air Force Office of Scientific Research under grant number FA8655-09-1-3055.