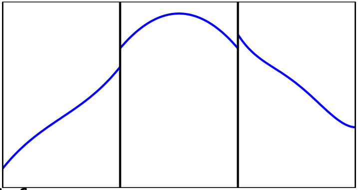
Local P-adaptivity with the Discontinuous Galerkin Method

Devin Light
Scott Moe

March 13th, 2015

1D DG-FEM

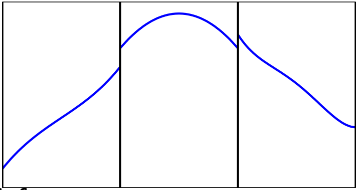


Conservation law weak form:

- Multiply by test function:

$$\frac{1}{2}\int_{-1}^{1} \varphi^{(\ell)} \left\{ q_{,t} + f(q)_{,x} \right\} d\xi = 0$$

1D DG-FEM



Conservation law weak form:

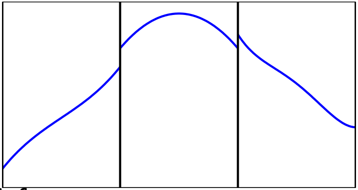
- Multiply by test function:

$$\frac{1}{2}\int_{-1}^{1}\varphi^{(\ell)}\left\{q_{,t}+f(q)_{,x}\right\}d\xi=0$$

- Integrate by parts

$$\frac{1}{2}\int_{-1}^{1}\varphi^{(\ell)}q_{,t}\,d\xi=\frac{1}{\Delta x}\left(\int_{-1}^{1}f\left(q^{h}\right)\,\varphi_{\xi}^{(k)}\,d\xi-\left[\left(\varphi^{(k)}\,f(q)\right)|_{\xi=1}-\left(\varphi^{(k)}\,f(q)\right)|_{\xi=-1}\right]\right)$$

1D DG-FEM



Conservation law weak form:

- Multiply by test function:

$$\frac{1}{2}\int_{-1}^{1} \varphi^{(\ell)} \left\{ q_{,t} + f(q)_{,x} \right\} d\xi = 0$$

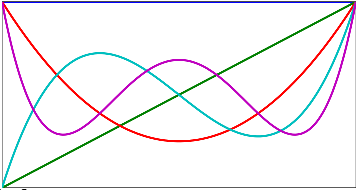
- Integrate by parts

$$\frac{1}{2}\int_{-1}^{1} \varphi^{(\ell)} q_{,t}\, d\xi = \frac{1}{\Delta x}\left(\int_{-1}^{1} f\left(q^h\right) \varphi_{\xi}^{(k)}\, d\xi - \left[\left(\varphi^{(k)} f(q)\right)|_{\xi=1} - \left(\varphi^{(k)} f(q)\right)|_{\xi=-1} \right] \right)$$

- Galerkin ansatz:

$$q(t,\xi) \sim \sum_{k=0}^{N} Q^{(k)}(t)\, \varphi^{(k)}(\xi)$$

1D DG-FEM

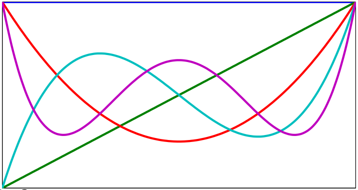


Conservation law weak form:

- Use an Orthogonal Basis:

$$\frac{1}{2}\int_{-1}^{1}\varphi^{(\ell)} q_{,t} = \frac{1}{2k+1}\dot{Q}^{(k)}$$

1D DG-FEM



Conservation law weak form:

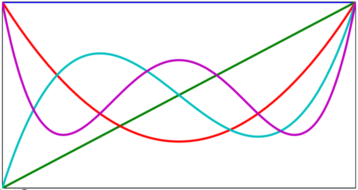
- Use an Orthogonal Basis:

$$\frac{1}{2}\int_{-1}^{1}\varphi^{(\ell)}q_{,t} = \frac{1}{2k+1}\dot{Q}^{(k)}$$

- Semi-discrete:

$$\dot{Q}_i^{(k)} = \frac{2k+1}{\Delta x}\int_{-1}^{1} f\left(q^h\right)\,\varphi_\xi^{(k)}\,d\xi - \frac{2k+1}{\Delta x}\left[\varphi^{(k)}(1)\,F_{i+\frac{1}{2}} - \varphi^{(k)}(-1)\,F_{i-\frac{1}{2}}\right]$$

1D DG-FEM



Conservation law weak form:

- Use an Orthogonal Basis:

$$\frac{1}{2}\int_{-1}^{1}\varphi^{(\ell)}q_{,t} = \frac{1}{2k+1}\dot{Q}^{(k)}$$

- Semi-discrete:

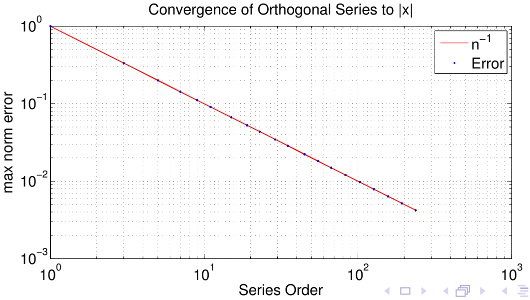
$$\dot{Q}_i^{(k)} = \frac{2k+1}{\Delta x}\int_{-1}^{1} f\left(q^h\right)\varphi_\xi^{(k)}\,d\xi - \frac{2k+1}{\Delta x}\left[\varphi^{(k)}(1)\,F_{i+\frac{1}{2}} - \varphi^{(k)}(-1)\,F_{i-\frac{1}{2}}\right]$$

- For $k=0$:

$$\dot{Q}_i^{(0)} = -\frac{1}{\Delta x}\left[F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}\right]$$

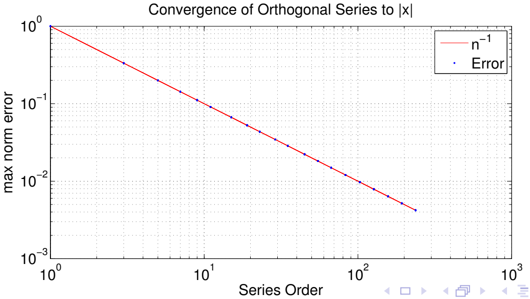
Orthogonal Series

- For continuous functions series coefficients converge.
- Coefficients give us a proxy for error.



Orthogonal Series

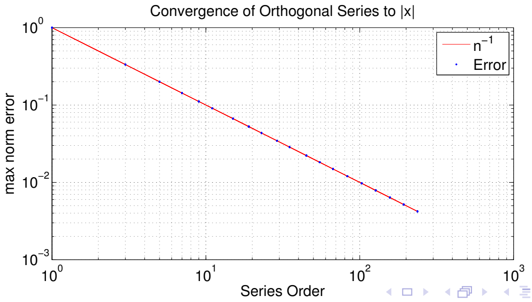
- For continuous functions series coefficients converge.
- Coefficients give us a proxy for error.
- Small coefficients have little effect on our solution.



Orthogonal Series

- For continuous functions series coefficients converge.
- Coefficients give us a proxy for error.
- Small coefficients have little effect on our solution.
- If f has a derivative ν of bounded variation

$$\|f - f_p\| \leq \sum_{i=p+1}^{\infty} |c_k \varphi^{(k)}(\xi)| \leq \frac{1}{\nu(p-\nu)^{\nu}}$$



hp-Adaptivity

- Discretize domain with cells of size h
- h-adaptivity $\rightarrow$ Adaptive Mesh Refinement

hp-Adaptivity

- Discretize domain with cells of size h
- h-adaptivity $\rightarrow$ Adaptive Mesh Refinement
- Discretize using piecewise continuous polynomials of order p.
- p-adaptivity $\rightarrow$ Add higher order polynomials:

$$q(t,\xi) = \sum_{k=0}^{p} Q^{(k)}(t)\,\varphi^{(k)}(\xi)$$

hp-Adaptivity

- Discretize domain with cells of size h
- h-adaptivity $\rightarrow$ Adaptive Mesh Refinement
- Discretize using piecewise continuous polynomials of order p.
- p-adaptivity $\rightarrow$ Add higher order polynomials:

$$q(t,\xi) = \sum_{k=0}^{p} Q^{(k)}(t)\,\varphi^{(k)}(\xi)$$

- For smooth functions:
  higher order polynomials $\rightarrow$ **MUCH** better approximation

Constructing an Algorithm

- For DG p can be different on each cell

Constructing an Algorithm

- For DG p can be different on each cell
- The highest coefficient gives an indication of error on each cell
- Each coefficient requires extra work

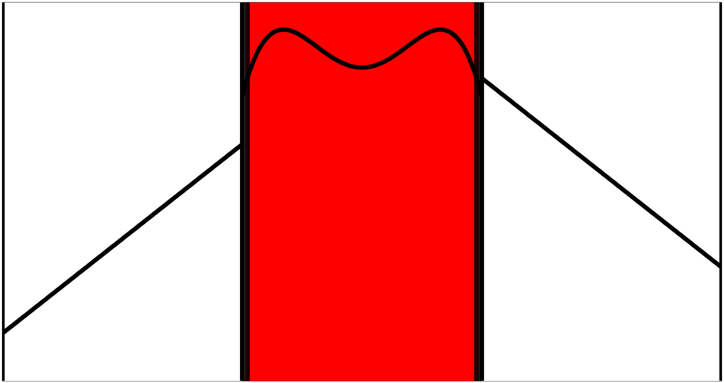
Constructing an Algorithm

- For DG p can be different on each cell
- The highest coefficient gives an indication of error on each cell
- Each coefficient requires extra work
- Idea:
  1. Very small coefficients $\rightarrow$ can be ignored
  2. Large coefficients $\rightarrow$ need more resolution

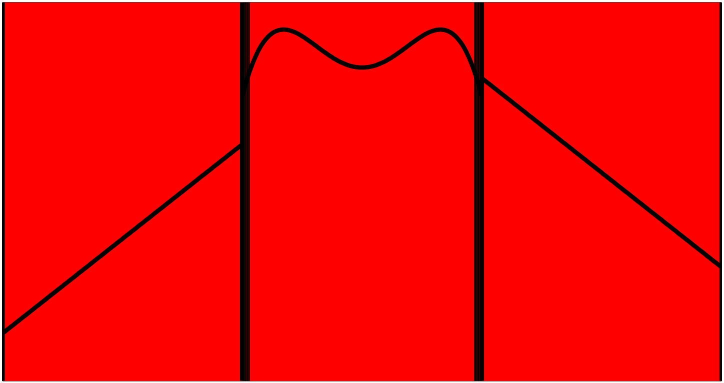
Constructing an Algorithm

- For DG p can be different on each cell
- The highest coefficient gives an indication of error on each cell
- Each coefficient requires extra work
- Idea:
  1. Very small coefficients $\rightarrow$ can be ignored
  2. Large coefficients $\rightarrow$ need more resolution
- This is complicated by the PDE
  1. All PDEs propagate information
  2. Hyperbolic PDEs $\rightarrow$ propagation speed is finite
  3. Cells with high-order neighbors may soon become high-order

Buffer Regions



Buffer Regions



Outline of the Algorithm

Pick a maximum order p_M and project initial conditions

Our p-adaptive algorithm:

Outline of the Algorithm

Pick a maximum order p_M and project initial conditions

Our p-adaptive algorithm:

1. Eliminate coefficients below a tolerance ϵ

Outline of the Algorithm

Pick a maximum order p_M and project initial conditions

Our p-adaptive algorithm:

1. Eliminate coefficients below a tolerance ϵ
2. Determine time-interval between local order modifications
3. Set buffer region (determined from characteristic speeds)

Outline of the Algorithm

Pick a maximum order p_M and project initial conditions

Our p-adaptive algorithm:

1. Eliminate coefficients below a tolerance ϵ
2. Determine time-interval between local order modifications
3. Set buffer region (determined from characteristic speeds)
4. Evolve coefficients over the interval

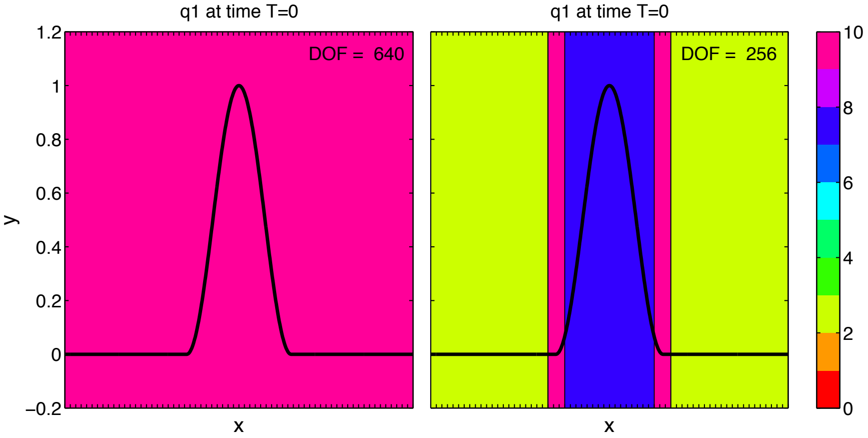
Outline of the Algorithm

Pick a maximum order p_M and project initial conditions

Our p-adaptive algorithm:

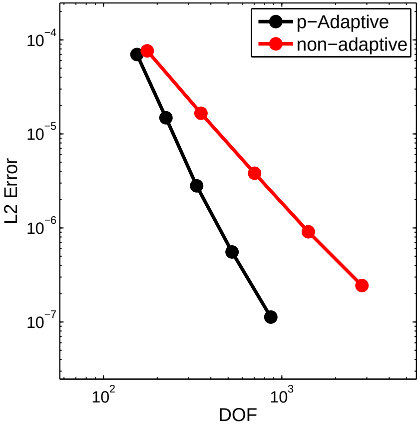
1. Eliminate coefficients below a tolerance ϵ
2. Determine time-interval between local order modifications
3. Set buffer region (determined from characteristic speeds)
4. Evolve coefficients over the interval
5. Repeat until we have reached the final time

Cosbell Advection



Cosbell Advection

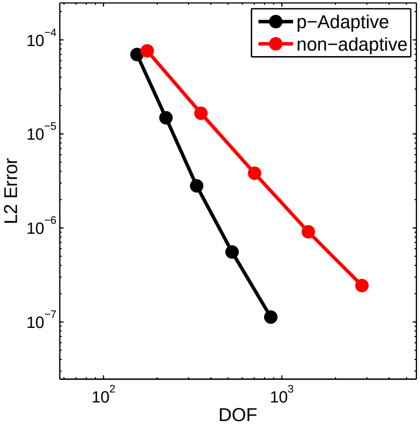
Both methods similarly accurate



Cosbell Advection

Both methods similarly accurate

p-Adaptive uses less DOFs!

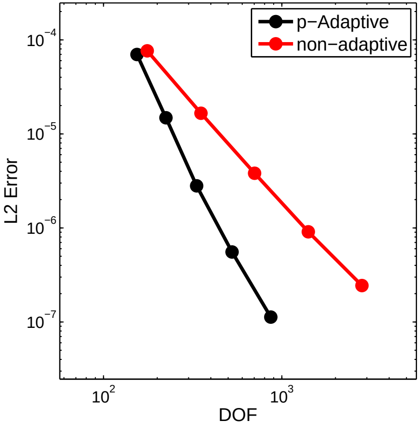


Cosbell Advection

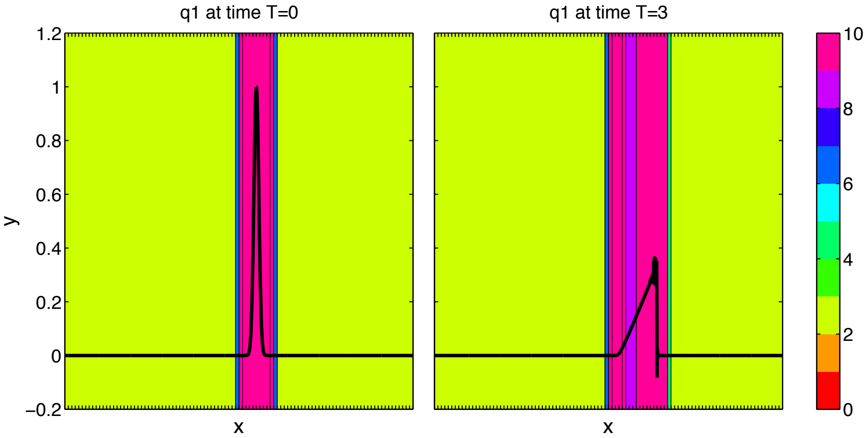
Both methods similarly accurate

p-Adaptive uses less DOFs!

Less DOFs $\rightarrow$ Less CPU time



Burger's Equation



Conclusions

1. Modifying local polynomial degree can be an efficient method of adding/removing DOFs in problems which demand efficiency
   - Atmospheric modeling, Seismology, etc..

Conclusions

1. Modifying local polynomial degree can be an efficient method of adding/removing DOFs in problems which demand efficiency
   - Atmospheric modeling, Seismology, etc..
2. Works best where solution is smooth
   - Coefficients decay quickly

Conclusions

1. Modifying local polynomial degree can be an efficient method of adding/removing DOFs in problems which demand efficiency
   - Atmospheric modeling, Seismology, etc..
2. Works best where solution is smooth
   - Coefficients decay quickly
3. Not as great where solutions are discontinuous
   - Coefficients don't decay at all

The Future

1. Local high order time stepping (Local Lax-Wendroff method)
2. Simultaneous h–p adaptivity
3. Limiting?
4. Adaptive quadrature

References

[1] J.-F. Remacle, J. E. Flaherty, and M. S. Shephard, "An adaptive discontinuous galerkin technique with an orthogonal basis applied to compressible flow problems," *SIAM review*, vol. 45, no. 1, pp. 53–72, 2003.

[2] M. J. Berger and R. J. LeVeque, "Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems," *SIAM Journal on Numerical Analysis*, vol. 35, no. 6, pp. 2298–2316, 1998.

[3] C. Eskilsson, "An hp-adaptive discontinuous galerkin method for shallow water flows," *International Journal for Numerical Methods in Fluids*, vol. 67, no. 11, pp. 1605–1623, 2011.

[4] G. Tumolo, L. Bonaventura, and M. Restelli, "A semi-implicit, semi-lagrangian, p-adaptive discontinuous galerkin method for the shallow water equations," *Journal of Computational Physics*, vol. 232, no. 1, pp. 46–67, 2013.

Thank you!

Questions?