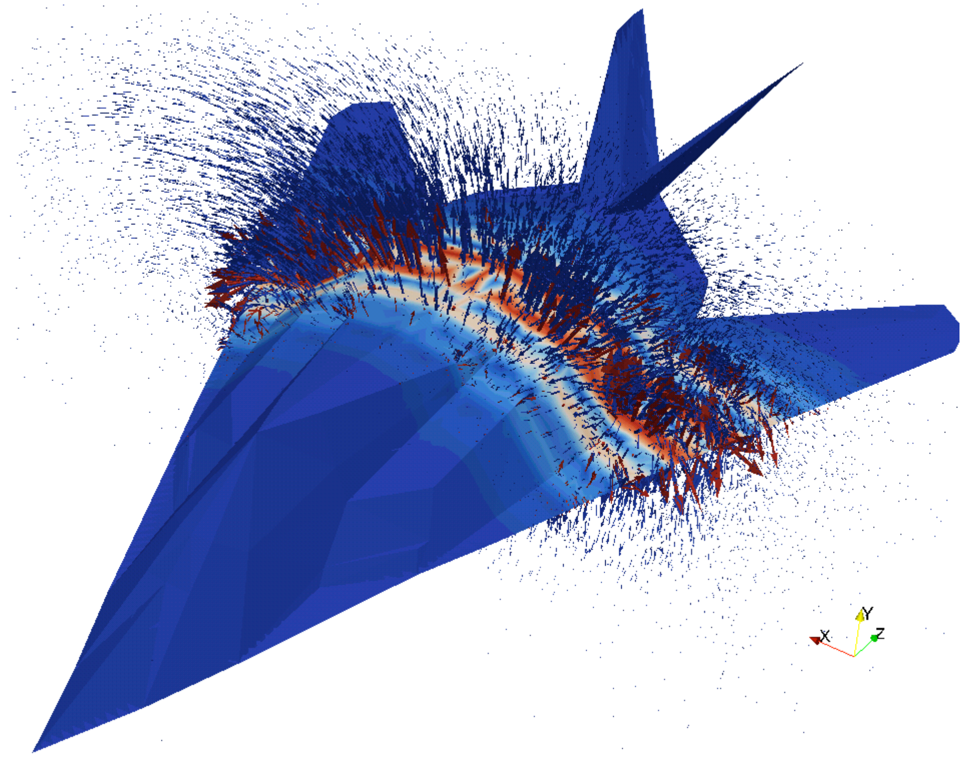
INRIA Sophia-Antipolis, 2009

Accelerating high-order accurate computational methods for solving PDE's

Jan S Hesthaven
Brown University
Jan.Hesthaven@Brown.edu



Thanks !

- ✓ Stephane Lanteri
- ✓ You for taking the time !
- ✓ Collaborators:
  - Tim Warburton (Rice)
  - Andreas Kloeckner (Brown)
  - Akil Narayan (Brown)
  - Lucas Wilcox (ICES, UT Austin)
  - Nico Godel (Hamburg, Germany)
- ✓ Funding agencies:
  - AFOSR, NSF, Nvidia

Key challenge

Central challenge in many computational modeling and design efforts

<u>Computational time</u>

This is caused by

- ✓ Large problems
- ✓ Non-linearity
- ✓ Open domains
- ✓ Requirement for high accuracy
- ✓ Long time integration
- ✓ Small cells

Overview of talk

Three different ways to combat this problem

✓ Recall DG-FEM

✓ Part I: A new basis well suited for open domains

✓ Part II: Local time-stepping

✓ Part III: GPU acceleration of DG-FEM

Mistake - several talks in one - Sorry !

Recall DG-FEM (for EM)

Consider Maxwell's equations

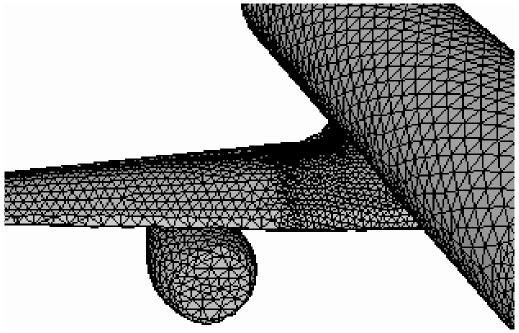
$$\varepsilon \partial_t E - \nabla \times H = -j, \qquad \mu \partial_t H + \nabla \times E = 0,$$

Write it on conservation form as

$$\frac{\partial q}{\partial t} + \nabla \cdot F = -J \quad F = \begin{bmatrix} -\hat{e} \times H \\ \hat{e} \times E \end{bmatrix} \quad q = \begin{bmatrix} E \\ H \end{bmatrix}$$

Represent the solution as

$$\Omega = \sum_k D^k \quad q_N = \sum_{i=1}^{N} q(\mathbf{x}_i, t) L_i(\mathbf{x})$$



and assume

$$\int_D \left(\frac{\partial \boldsymbol{q}_N}{\partial t} + \nabla \cdot \boldsymbol{F}_N - \boldsymbol{J}_N \right) L_i(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = \oint_{\partial D} L_i(\boldsymbol{x}) \hat{\boldsymbol{n}} \cdot [\boldsymbol{F}_N - \boldsymbol{F}^*]\, \mathrm{d}\boldsymbol{x}.$$

Recall DG-FEM (for EM)

On each element we then define

$$\hat{M}_{ij} = \int_D L_i L_j \, \mathrm{d}\boldsymbol{x}, \quad \hat{S}_{ij} = \int_D \nabla L_j L_i \, \mathrm{d}\boldsymbol{x}, \quad \hat{F}_{ij} = \oint_{\partial D} L_i L_j \, \mathrm{d}\boldsymbol{x},$$

With the numerical flux given as

$$\hat{\boldsymbol{n}} \cdot [\boldsymbol{F} - \boldsymbol{F}^*] = \begin{cases} \boldsymbol{n} \times (\gamma \boldsymbol{n} \times [\boldsymbol{E}] - [\boldsymbol{B}]), \\ \boldsymbol{n} \times (\gamma \boldsymbol{n} \times [\boldsymbol{B}] + [\boldsymbol{E}]), \end{cases} \qquad [Q] = Q^- - Q^+$$

To obtain the local matrix based scheme

$$\boxed{\hat{M} \frac{\mathrm{d}\hat{\boldsymbol{q}}}{\mathrm{d}t} + \hat{S} \cdot \hat{\boldsymbol{F}} - \hat{M}\hat{\boldsymbol{J}} = \hat{F}\hat{\boldsymbol{n}} \cdot [\hat{\boldsymbol{F}} - \hat{\boldsymbol{F}}^*],}$$

One then typically uses an explicit Runge-Kutta or a LeapFrog method to advance in time

Recall DG-FEM

The advantages of this approach are many and the scheme is well understood :

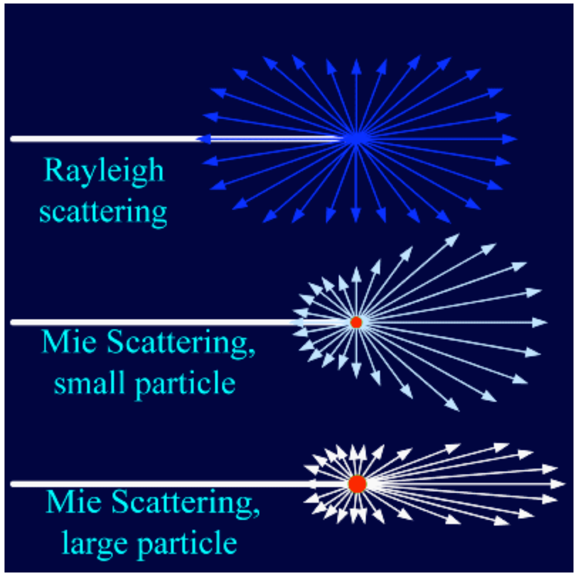
- ✓ High-order, geometrically flexible, robust, explicit etc
- ✓ Well understood
- ✓ Generalizes to broad class of problems

... but a central criticism is *speed* - or lack of it !

Part I: Unbounded problems

The need to numerically solve problems on semi-infinite/infinite domains arises in many applications:

- ➡ Acoustic/Electromagnetic/Elastic scattering
- ➡ Kinetic/Boltzmann models
- ➡ Computational chemistry
- ➡ Molecular dynamics
- ➡ Numerical relativity
- ➡ etc



Introduction

- Approximate/absorbing boundary conditions
  - Typically problem dependent

- Domain truncation
  - Where to truncate ?

- Infinite expansions
  - Hermite/Laguerre polynomials/functions
    - Expensive/inflexible - require exp(-|x|)
    - .... but O(N) spectrum
  - Rational/mapped Chebyshev methods (Boyd)
    - Amenable to FFT
    - .... but O(N*N) spectrum

Objective

What we seek is a new basis set with the properties

- ➡ Controllable asymptotic decay of basis
- ➡ The FFT can be used to evaluate
- ➡ The spectrum is O(N) for 1st order operator

.. but is it possible ?

Motivation - Wiener('49) proposed the rational basis

$$\phi_n(x) = \frac{(1-ix)^n}{\sqrt{\pi}(1+ix)^{n+1}}, \qquad n \in \mathbb{N}_0 \qquad \propto \frac{1}{|x|}, \quad |x| \to \infty$$

- ➡ Orthonormal (and can be made complete)
- ➡ Fourier transform of Laguerre functions

Some previous work

Several authors have considered this basis

➡Higgins (1977) considered even/odd real basis and proved L2-completeness of complex basis

➡Christov (1982 and later) extended some of this and also applied the basis to solve PDE's

➡Boyd (1990) offers some comparison with mapped functions

➡Weideman (1992) consider basic properties of operators

Let's sketch how this is possible ...

Several of the requirements suggest we take off from the Fourier basis

$$\psi_k(\theta) = e^{ik\theta}.$$

Rewrite this as (Szego'30)

$$\begin{aligned}
e^{ik\theta} &= \cos(k\theta) &+& \quad i\sin(k\theta) \\
&= \cos(|k|\,\theta) &+& \quad i\,\mathrm{sgn}(k)\sin(|k|\,\theta) \\
&= T_{|k|}(\cos\theta) &+& \quad i\,\mathrm{sgn}(k)\sin(\theta)\,U_{|k|-1}(\cos\theta) \\
&= \sqrt{\frac{\pi}{2}}\Big[\tilde{P}^{(-1/2,-1/2)}_{|k|}(\cos\theta) &+& \quad i\,\mathrm{sgn}(k)\ \sin(\theta)\,\tilde{P}^{(1/2,1/2)}_{|k|-1}(\cos\theta)\Big].
\end{aligned}$$

Even Odd

Let's sketch how this is possible ...

Can we generalize the Fourier basis by combining Jacobi polynomials in a special way:

➡ Maintain orthogonality of the basis

➡ Maintain connection to Fourier basis for FFT

Szego solved it (at least in spirit)

Theorem 2.2. (Szegö, [3]) *For any* $\gamma > -\frac{1}{2}$, *the functions*

$$\Psi_k^{(\gamma)}(\theta) = \begin{cases} \frac{1}{\sqrt{2}} \tilde{P}_0^{(-1/2,\gamma-1/2)}(\cos\theta), & k=0 \\ \frac{1}{2}\Big[\tilde{P}_{|k|}^{(-1/2,\gamma-1/2)}(\cos\theta) + i\,\mathrm{sgn}(k)\sin(\theta)\tilde{P}_{|k|-1}^{(1/2,\gamma+1/2)}(\cos\theta)\Big], & k \neq 0 \end{cases}$$

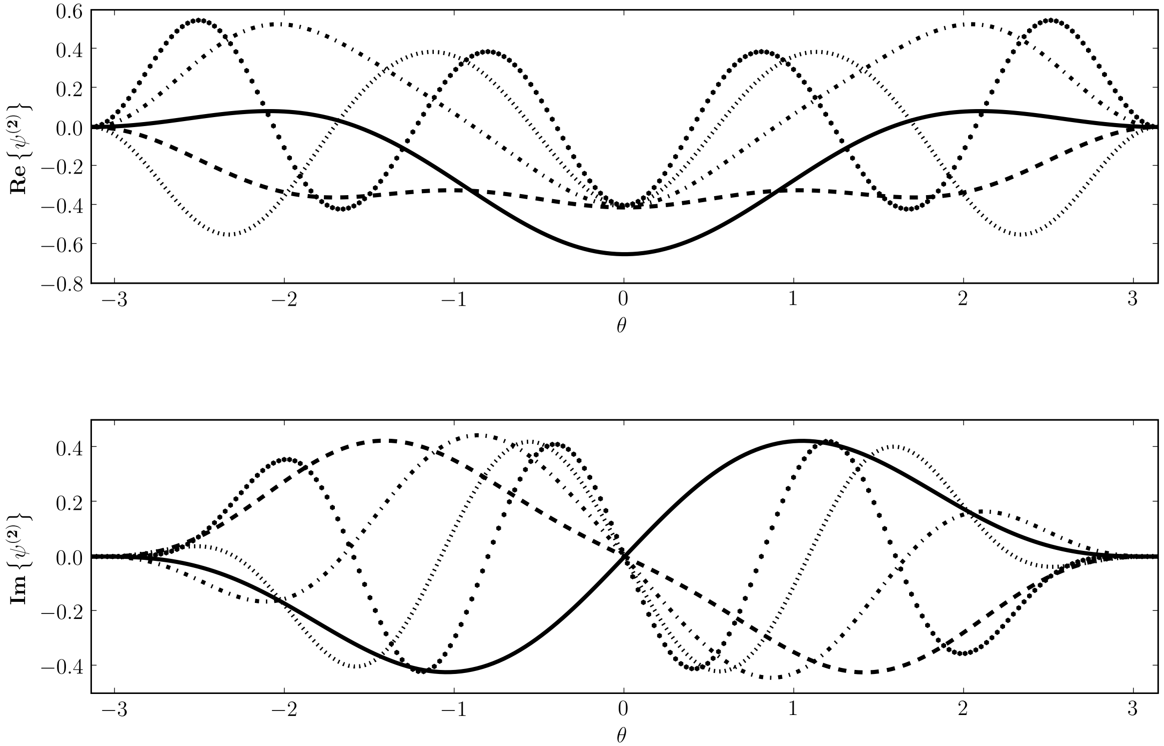
are complete and orthonormal in $L^2\Big([-\pi,\pi],\mathbb{C}; w_\theta^{(\gamma,0)}\Big)$.

$$w_\theta^{(\gamma,\delta)}(\theta) \;=\; w_r^{(\delta,\gamma)}(r(\theta)) = (1+\cos\theta)^\gamma(1-\cos\theta)^\delta$$

Let’s sketch how this is possible ...

Let’s orthonormalize them

$$\psi_k^{(\gamma)}(\theta) = \begin{cases} \dfrac{\sqrt[*]{w_\theta^{(\gamma,0)}}}{\sqrt{2}} \tilde{P}_0^{(-1/2,\gamma-1/2)}(\cos\theta), & k=0 \\ \dfrac{\sqrt[*]{w_\theta^{(\gamma,0)}}}{2} \Big[\tilde{P}_{|k|}^{(-1/2,\gamma-1/2)}(\cos\theta) + i\,\mathrm{sgn}(k)\sin(\theta)\tilde{P}_{|k|-1}^{(1/2,\gamma+1/2)}(\cos\theta) \Big], & k \neq 0 \end{cases}$$



Note: Decay as

$$\boxed{\left(\cos\frac{\theta}{2}\right)^{\gamma}}$$

for

$$\theta \to \pm\pi$$

Let's sketch how this is possible ...

Taking it to the unbounded domain involves

$$\cos\theta = \frac{1-x^2}{1+x^2}, \qquad (1-\cos\theta) = \frac{2x^2}{x^2+1},$$

$$\sin\theta = \frac{2\,x}{x^2+1}, \qquad (1+\cos\theta) = \frac{2}{x^2+1}.$$

Leading to

$$\boxed{\gamma = s-1}$$

$$\Phi_k^{(s)}(x) := \Psi_k^{(s-1)}(\theta)$$

$$= \begin{cases} \dfrac{1}{\sqrt{2}}\tilde{P}_0^{(-1/2,s-3/2)}\left(\dfrac{1-x^2}{1+x^2}\right), & k=0 \\ \dfrac{1}{2}\left[\tilde{P}_{|k|}^{(-1/2,s-3/2)}\left(\dfrac{1-x^2}{1+x^2}\right) + \dfrac{2\,ix\,\mathrm{sgn}(k)}{x^2+1}\tilde{P}_{|k|-1}^{(1/2,s-1/2)}\left(\dfrac{1-x^2}{1+x^2}\right)\right], & k\neq 0 \end{cases}$$

Note: Still Chebyshev-like Jacobi polynomials

Let's sketch how this is possible ...

The orthonormal basis is

$$\phi_k^{(s)} \;:=\; \sqrt[*]{w_x^{(s,0)}}\,\Phi_k^{(s)}(x)$$

$$= \begin{cases} \dfrac{2^{\left(\frac{s-1}{2}\right)}}{(x-i)^s}\tilde{P}_0^{(-1/2,s-3/2)}\left(\dfrac{1-x^2}{1+x^2}\right), & k=0 \\[2ex] \dfrac{2^{\left(\frac{s}{2}-1\right)}}{(x-i)^s}\left[\tilde{P}_{|k|}^{(-1/2,s-3/2)}\left(\dfrac{1-x^2}{1+x^2}\right)+\dfrac{2\,ix\,\mathrm{sgn}(k)}{x^2+1}\tilde{P}_{|k|-1}^{(1/2,s-1/2)}\left(\dfrac{1-x^2}{1+x^2}\right)\right], & k\neq 0. \end{cases}$$

Note:

$$i\sqrt{2}\,\phi_n^{(1)}(x) := \;\; \phi_n(x) = \frac{(1-ix)^n}{\sqrt{\pi}(1+ix)^{n+1}}, \qquad n\in\mathbb{N}_0$$

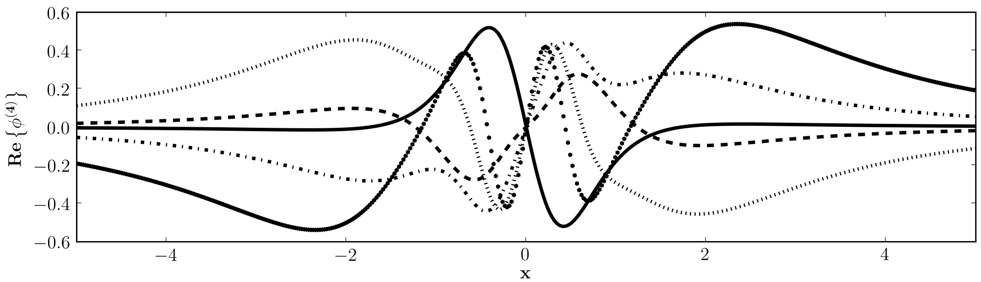
So we have ***generalized the Wiener rational basis***

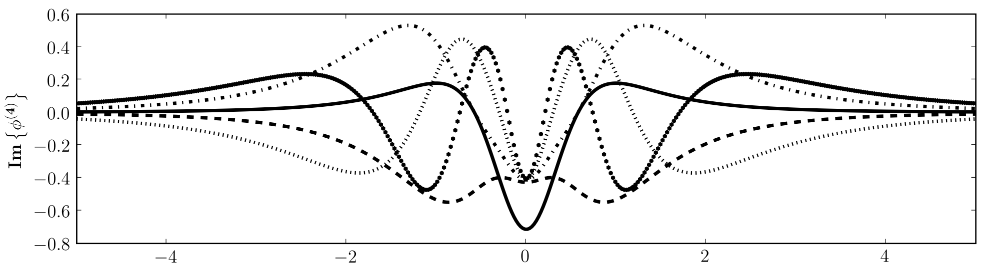
Let's sketch how this is possible ...

What about the decay rate ?

Proposition 2.5. *For any $s > \frac{1}{2}$, the functions $\Phi_k^{(s)}(x)$ are complete and orthonormal in $L^2\Big(\mathbb{R}, \mathbb{C}; w_x^{(s,0)}\Big)$. The functions $\phi_k^{(s)}(x)$ are complete and orthonormal in $L^2(\mathbb{R}, \mathbb{C})$. Furthermore, the decay rate of these functions can be characterized as*

$$\lim_{|x|\to\infty} \left| x^t \, \phi_k^{(s)}(x) \right| < \infty, \qquad t \le s$$





s=4, k=1..4

Parametrized decay rates

$$\phi_k^{(s)} \propto \frac{1}{|x|^s}, \quad |x| \to \infty$$

What about efficiency ?

Recall

$$\phi_k^{(s)} := \sqrt[*]{w_x^{(s,0)}}\, \Phi_k^{(s)}(x)$$

$$= \begin{cases} \dfrac{2^{\left(\frac{s-1}{2}\right)}}{(x-i)^s} \tilde{P}_0^{(-1/2,s-3/2)}\left(\dfrac{1-x^2}{1+x^2}\right), & k=0 \\[2ex] \dfrac{2^{(\frac{s}{2}-1)}}{(x-i)^s}\left[\boxed{\tilde{P}_{|k|}^{(-1/2,s-3/2)}\left(\dfrac{1-x^2}{1+x^2}\right)} + \dfrac{2\,ix\,\mathrm{sgn}(k)}{x^2+1}\tilde{P}_{|k|-1}^{(1/2,s-1/2)}\left(\dfrac{1-x^2}{1+x^2}\right)\right], & k\neq 0. \end{cases}$$

s=1: Chebyshev - FFT is possible

Recall also the connections

$$\tilde{P}_n^{(\alpha,\beta)} = \nu_{n,0}^{(\alpha,\beta)}\tilde{P}_n^{(\alpha+1,\beta)} - \nu_{n,-1}^{(\alpha,\beta)}\tilde{P}_{n-1}^{(\alpha+1,\beta)},$$

$$\tilde{P}_n^{(\alpha,\beta)} = \nu_{n,0}^{(\beta,\alpha)}\tilde{P}_n^{(\alpha,\beta+1)} + \nu_{n,-1}^{(\beta,\alpha)}\tilde{P}_{n-1}^{(\alpha,\beta+1)},$$

What about efficiency ?

We can clearly use the connection coefficients to connect the different families as

$$f(r) = \sum_{n=0}^{\infty} \hat{f}_n^{(\alpha,\beta)} \tilde{P}_n^{(\alpha,\beta)}(r) \quad \longrightarrow \quad f(r) = \sum_{n=0}^{\infty} \hat{f}_n^{(\alpha+A,\beta+B)} \tilde{P}_n^{(\alpha+A,\beta+B)}(r),$$

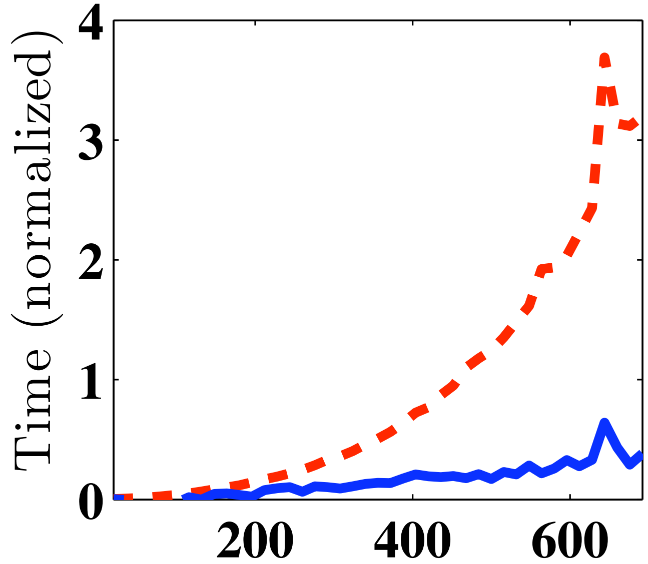
If (A,B) are integer one has the (non-trivial) result

$$\boxed{\hat{f}_n^{(\alpha+A,\beta+B)} = \sum_{m=0}^{A+B} \lambda^P_{n,n+m} \hat{f}_{n+m}^{(\alpha,\beta)}.}$$

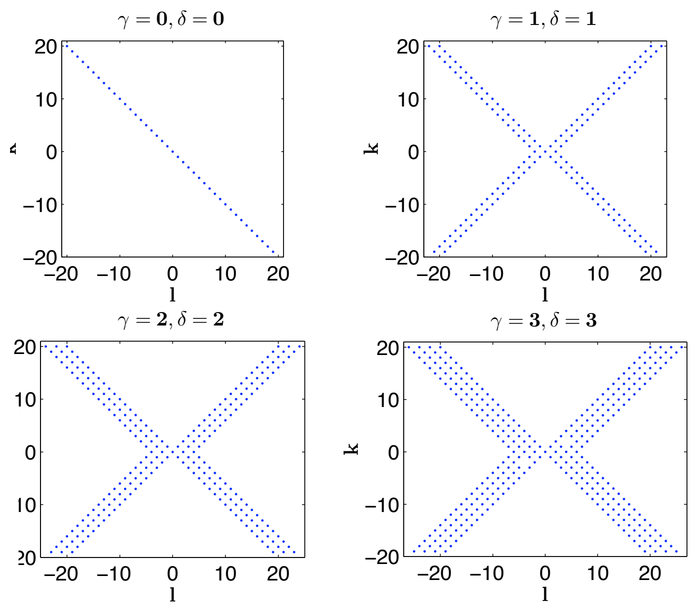
Note: One "could" compute connection coefficients directly -- but is better not to

What about efficiency ?

Using this to create connectivity operators, the FFT can be used to evaluate/manipulate the new basis



Connectivity operators are sparse



FFT speedup $\left(\frac{T_{direct}}{T_{fft}}\right)$

N	$s = 5$	$s = 6$	$s = 7$	$s = 8$	$s = 9$	$s = 10$
512	2.4	2.3	2.2	1.9	1.9	1.8
800	5.6	5.0	4.8	4.5	4.1	3.7
1024	8.1	7.1	7.0	6.4	6.1	5.5
1600	16.3	15.3	13.5	12.9	12.0	11.2
2025	23.2	21.2	20.0	17.4	16.8	15.9
2916	48.9	33.9	37.6	28.7	27.3	24.4

Cost scales like

$$\mathcal{O}(N \log N + (\gamma + 1)N)$$

Other basic properties of basis

- Simple convolution (for s=1 only)

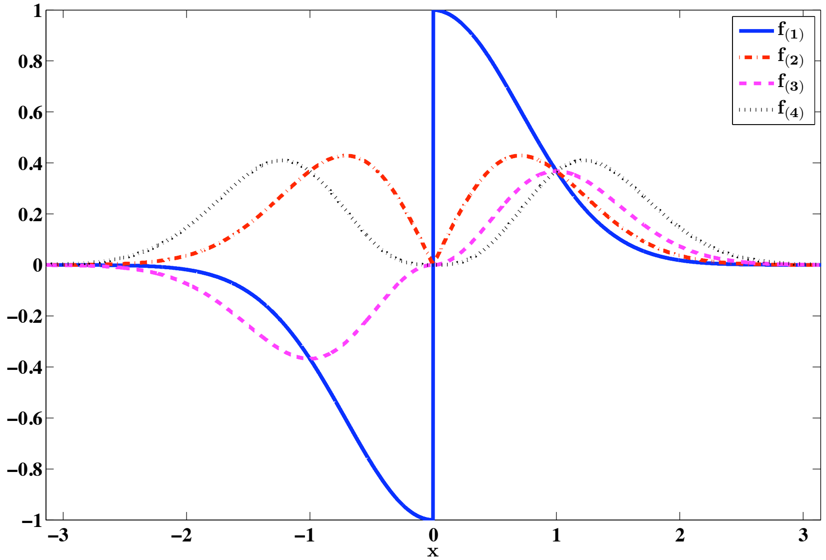
$$\phi_k^{(1)} \times \phi_l^{(1)} = \frac{1}{4\sqrt{\pi}} \left[\phi_{k+l+1}^{(1)} - \phi_{k+l}^{(1)} \right],$$

- Stiffness matrix is sparse and skew-symmetric

- Spectrum scales as N+Ks

s \ N	11	50	101	250	501
0.6	7.31	43.76	91.50	237.60	483.75
1.0	7.99	44.51	92.28	238.39	484.54
6.0	15.96	53.75	101.81	248.14	494.40
π^2	21.72	60.67	109.05	255.63	501.99
15.5	29.73	70.45	119.40	266.44	512.99

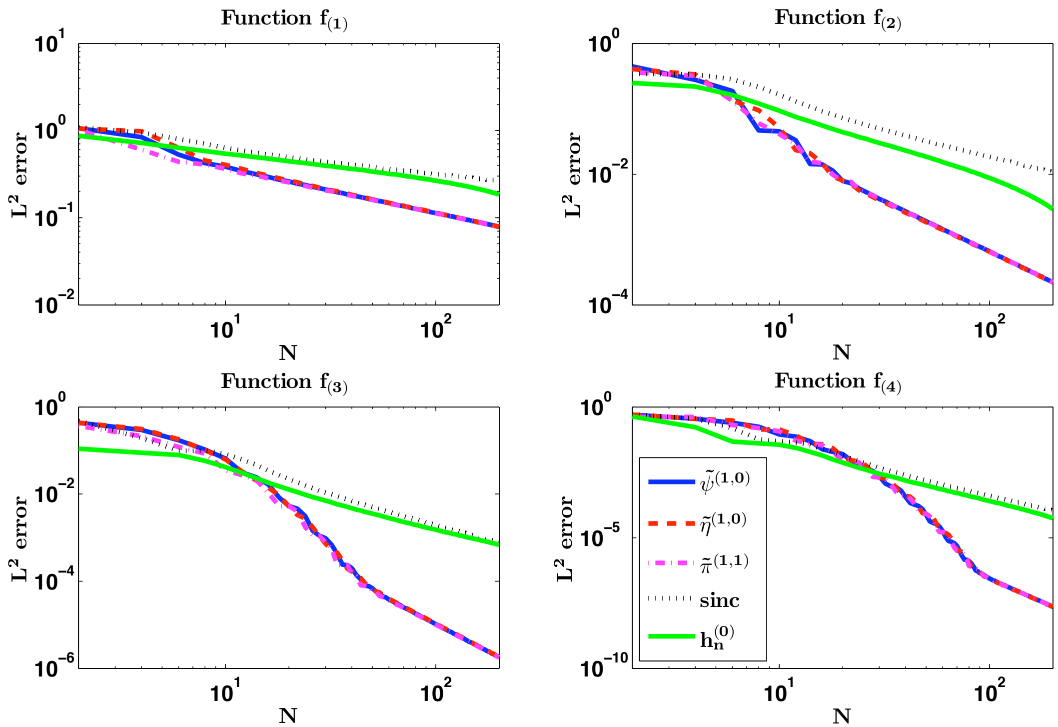
What about accuracy ?



Tests of increasing regularity

$$f_{(1)}(x) = \mathrm{sgn}(x)\, e^{-x^2}, \qquad f_{(2)}(x) = |x|\, e^{-x^2},$$

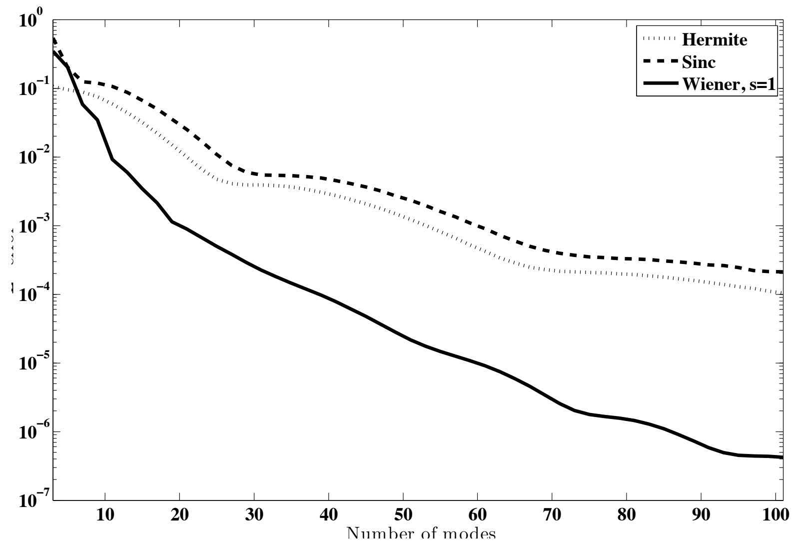
$$f_{(3)}(x) = \mathrm{sgn}(x)\, x^2 e^{-x^2}, \qquad f_{(4)}(x) = |x^3|\, e^{-x^2}.$$



Close relation between regularity and convergence rate as expected.

Approximation theory closely related to classic results

What about accuracy ?

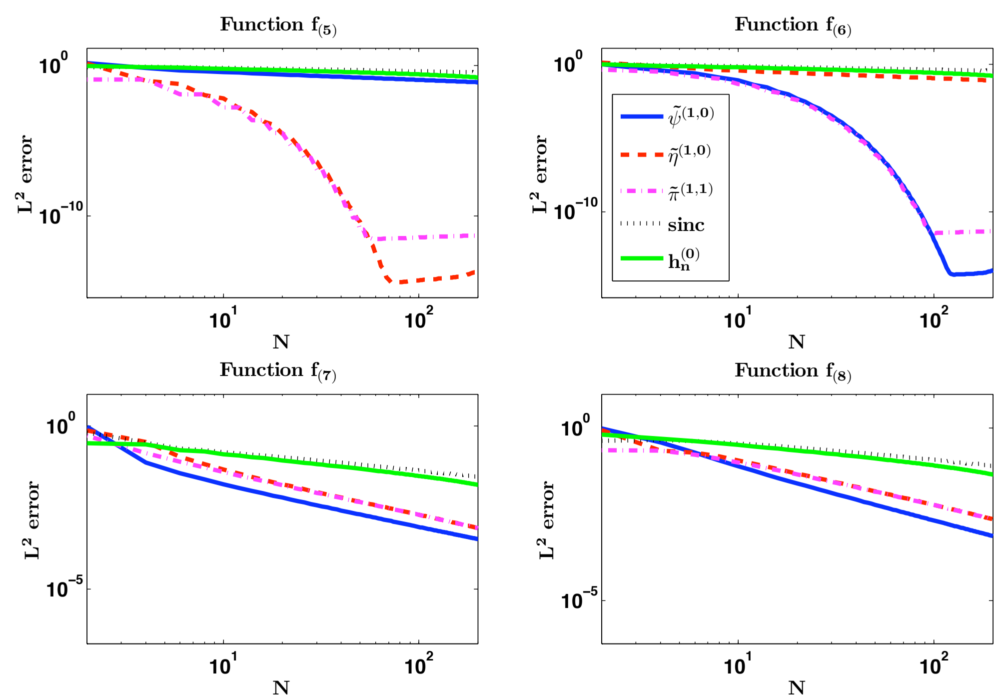


$$f(x) = \frac{\arctan(x+3)}{x^4+1},$$

Clearly superior to Hermite/Sinc

$$f_{(5)} = \frac{1}{\sqrt[4]{x^4+1}} \qquad f_{(6)} = \frac{x^5}{x^6+1}$$

$$f_{(7)} = \frac{1}{(x^2+1)^{7/8}} \qquad f_{(8)} = \frac{\log(x^2+2)}{x^2+1}.$$



Analysis is more involved here due to behavior at infinity

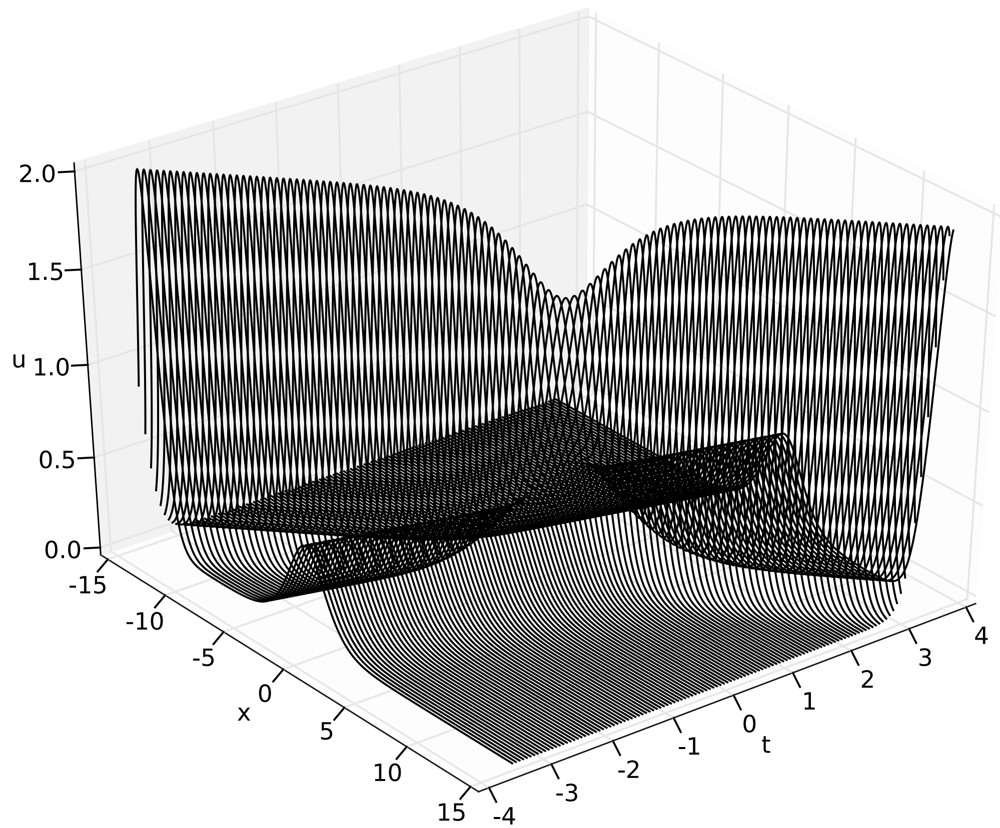
Example: Nonlinear Waves

We consider the 1D KdV equation

$$u_t + u_{xxx} + 6uu_x = 0, \qquad x \in \mathbb{R}$$

Exact 2-soliton solution

Exponential decay



Example: Nonlinear Waves

Total evolution time, $t = -3.5, \ldots, 3.5$

	N =50	N =100	N =150	N =200	N =300	N =400	N =500
Fourier	5.45e-01	4.53e+00	1.44e+01	3.47e+01	1.51e+02	3.92e+02	8.64e+02
Hermite	5.15e+00	4.88e+00	2.37e+01	7.05e+01	5.46e+02	2.13e+03	7.81e+03
Sinc	1.40e+00	2.31e+01	1.24e+02	4.63e+02	3.38e+03	——	——
Mapped Cheb.	8.90e-01	9.68e+00	3.72e+01	9.79e+01	3.60e+02	9.95e+02	2.65e+03
Wiener, $s = 1$	9.43e-01	9.70e+00	3.49e+01	8.88e+01	2.99e+02	7.25e+02	1.66e+03
Wiener, $s = 2$	2.06e+00	2.03e+01	7.45e+01	1.71e+02	5.34e+02	1.26e+03	2.81e+03
Wiener, $s = 5$	2.31e+00	2.33e+01	8.35e+01	1.91e+02	6.20e+02	1.51e+03	3.18e+03

L^2 errors

	N =50	N =100	order	N =150	order
Fourier	1.36e+00	2.43e-03	9.13	2.00e-03	0.474
Hermite	——	3.29e-02	——	2.12e-03	6.76
Sinc	4.71e-02	1.74e-04	8.08	1.74e-04	——
Mapped Cheb.	3.84e+00	5.74e-01	2.74	5.96e-02	5.59
Wiener, $s = 1$	3.54e+00	5.12e-01	2.79	5.57e-02	5.47

Example: Nonlinear Waves

Let's consider a slightly modified equation

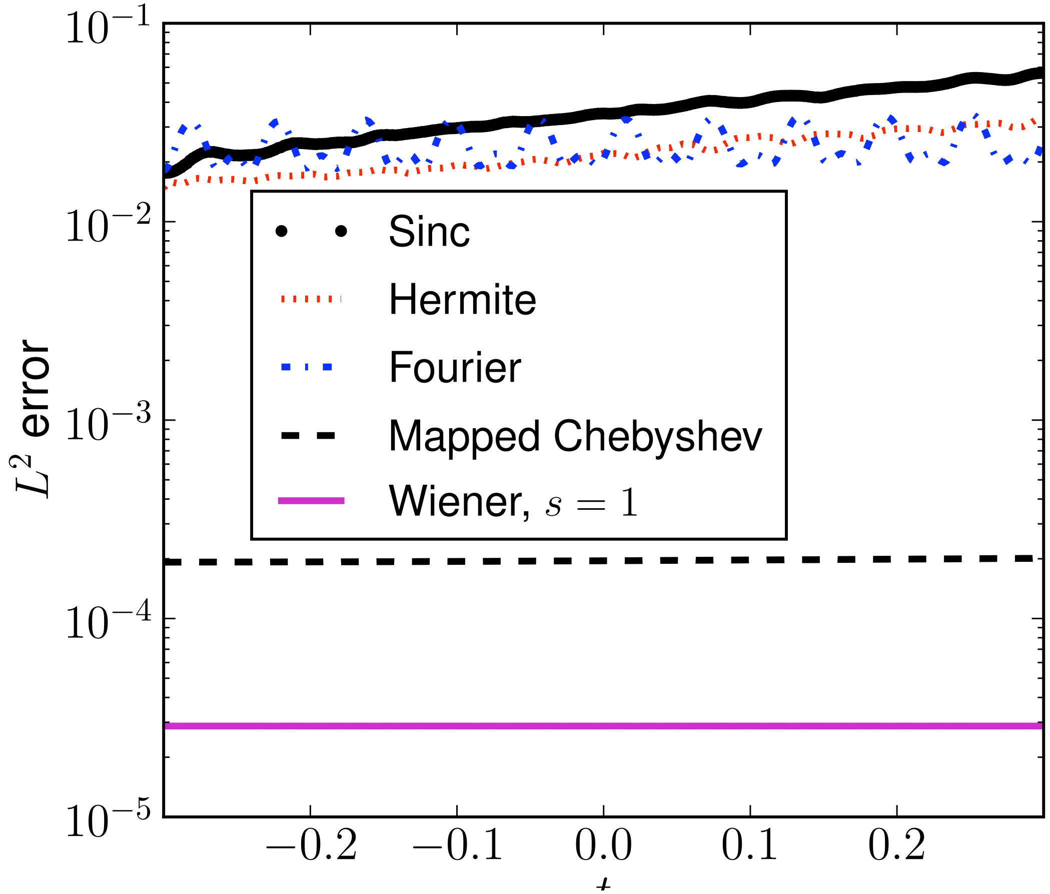
$$u_t + 6(u+1)^2 u_x + u_{xxx} = 0, \qquad x \in \mathbb{R}.$$

Solution

$$u(x,t) = \frac{-4}{4(x-6t)^2+1}.$$

Algebraic decay

N=150



Example: Vlasov equations

We consider the 1.5D consistent problem

$$\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} + \frac{q}{m}\left[(E_x + v_y B_z) \frac{\partial f}{\partial v_x} + (E_y - v_x B_z) \frac{\partial f}{\partial v_y} \right] = 0.$$

$$\frac{\partial E_x}{\partial t} = -\frac{1}{\varepsilon_0} J_x \qquad \frac{\partial B_z}{\partial t} + \frac{\partial E_y}{\partial x} = 0$$

$$\frac{\partial E_y}{\partial t} + c^2 \frac{\partial B_z}{\partial x} = -\frac{1}{\varepsilon_0} J_y \qquad \frac{\partial E_x}{\partial x} = \frac{\rho}{\varepsilon_0}$$

$$\begin{aligned} \rho(x,t) &= \int f(x,v,t)\, \mathrm{d}v_x\, \mathrm{d}v_y \\ J_x(x,t) &= \int v_x f(x,v,t)\, \mathrm{d}v_x\, \mathrm{d}v_y \\ J_y(x,t) &= \int v_y f(x,v,t)\, \mathrm{d}v_x\, \mathrm{d}v_y \end{aligned}$$

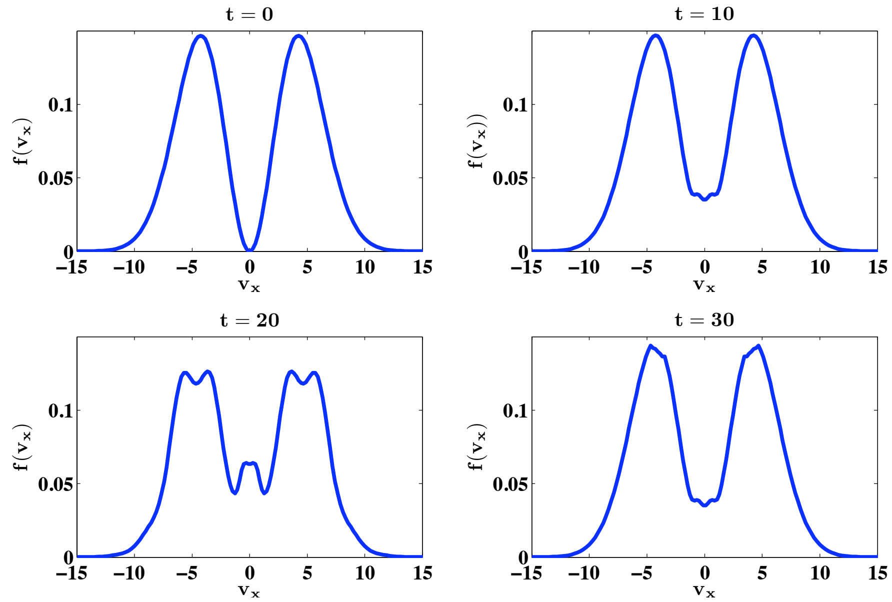
Problem in kinetic plasma physics

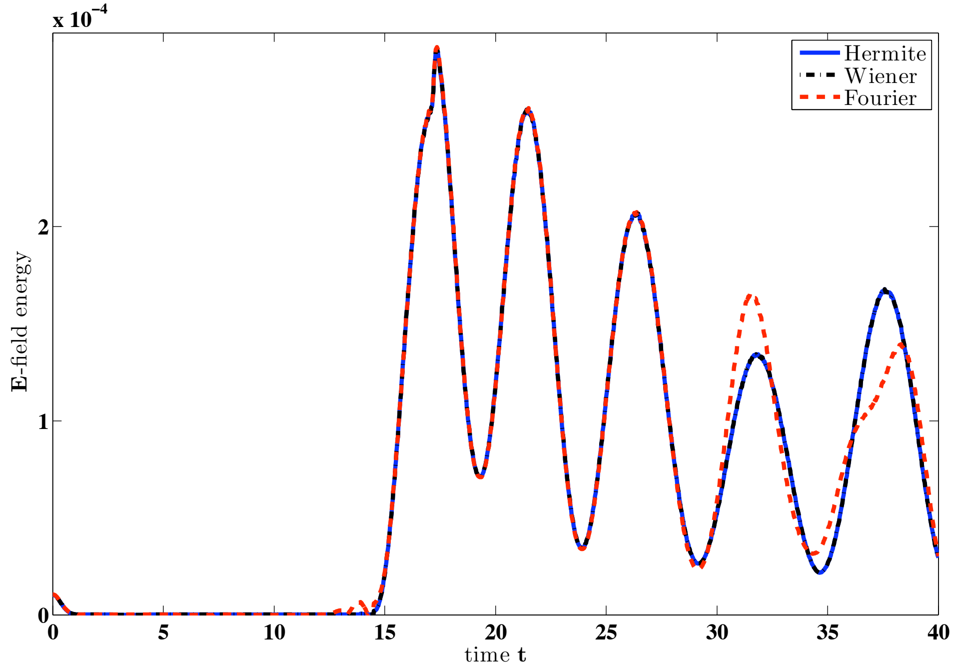
DG-FEM in physical space, Wiener expansion in velocity space

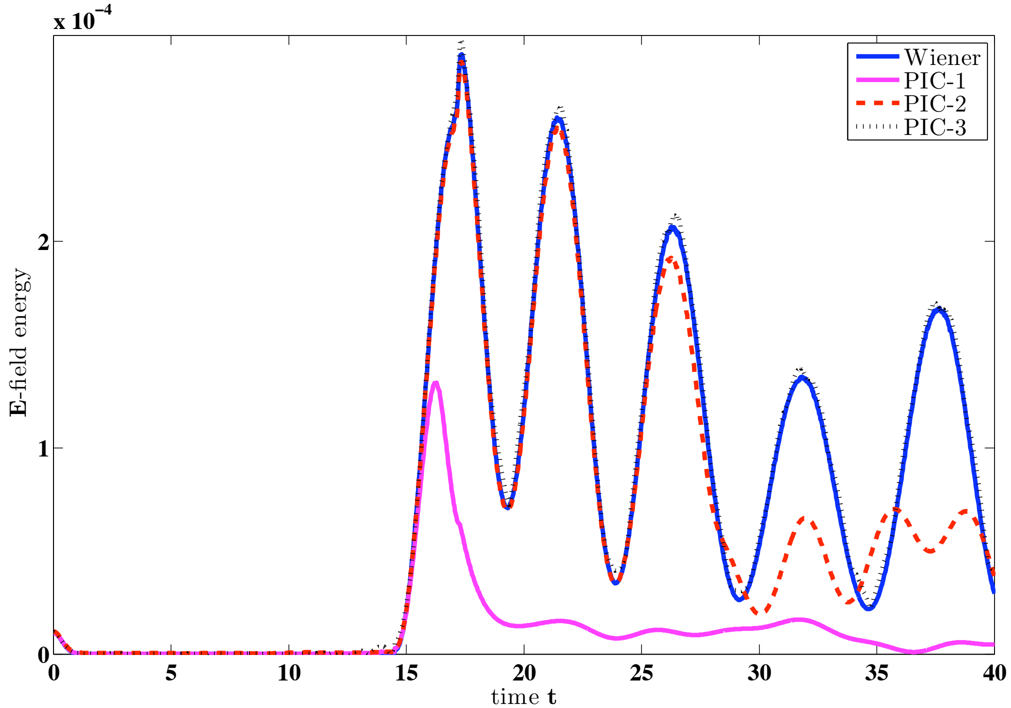
Example: Vlasov solvers

Consider a two-stream instability as test

$$f_0(x, v_x) = K v_x^2 \, e^{-v_x^2/2} \, (1 + \varepsilon \cos(\pi x)),$$







Example: Wave problem

Three dimensional wave problem

$$\tilde{u}_{tt} = c^2 \Delta \tilde{u}, \qquad (x,t) \in (\Gamma, [0,T]),$$

Assuming spherical symmetry yields *semi-infinite problem*

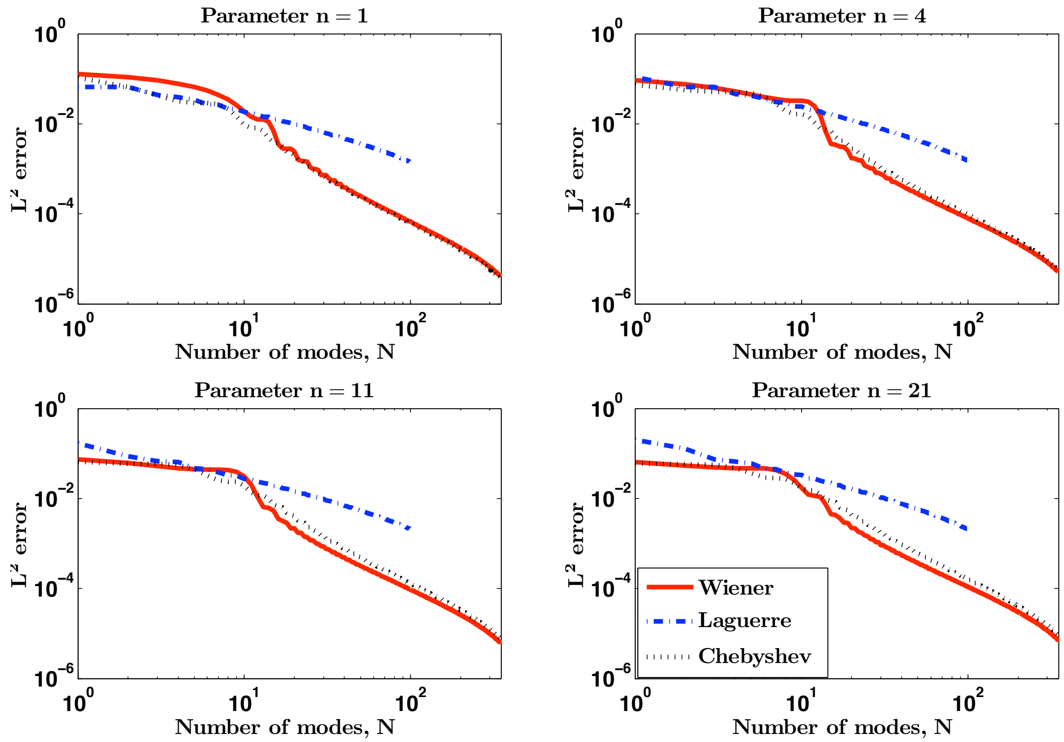
$$u_{tt} = c^2 \left[u_{\rho\rho} + \frac{2}{\rho} u_\rho - \frac{n(n+1)}{\rho^2} u \right].$$

With solution

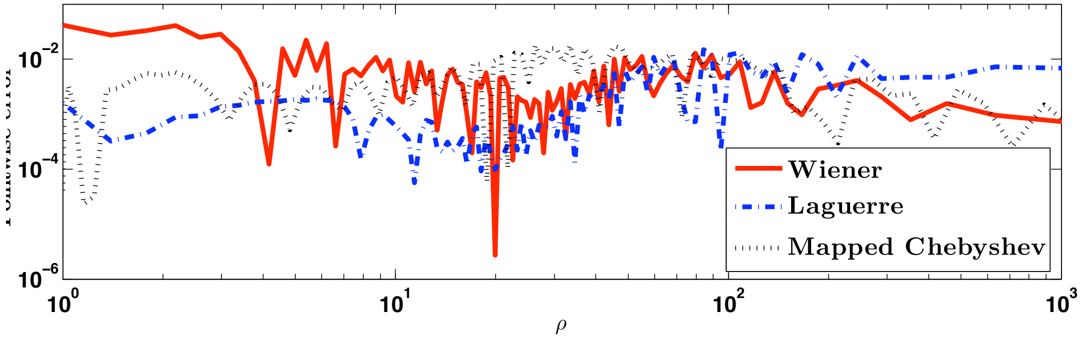
$$u(\rho,t) = \cos(ct)\hat{u}(\rho).$$

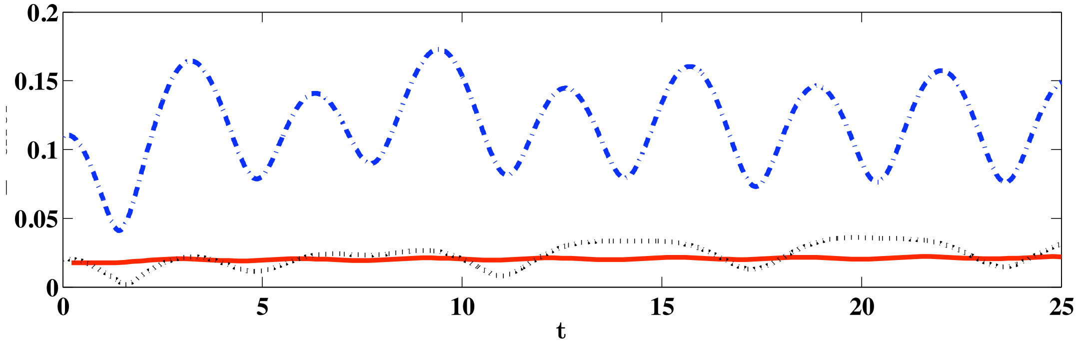
$$\hat{u}_{n,1}(\rho) \;=\; j_n(\rho) = \sqrt{\frac{\pi}{2\rho}} J_{n+1/2}(\rho)$$

$$\hat{u}_{n,2}(\rho) \;=\; y_n(\rho) = \sqrt{\frac{\pi}{2\rho}} Y_{n+1/2}(\rho),$$



Example: Wave problem





Mapped Chebychev and Wiener expansion clearly superior

Cost:

- Laguerre method: 391 sec
- Mapped Chebychev: 1019 sec
- Wiener method: 39 sec

Due to FFT and much larger time-step

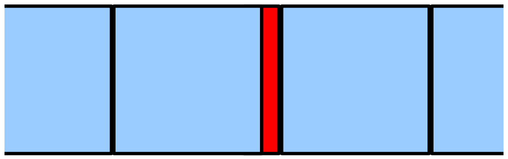
Summary on Part I

It seems that expansions based on these functions have interesting properties

- ➡ they are accurate
- ➡ the basis is flexible
- ➡ the evaluation is fast
- ➡ the spectral properties of operators are good
- ➡ other applications -- windowed Fourier series; basis for infinite FEM elements etc

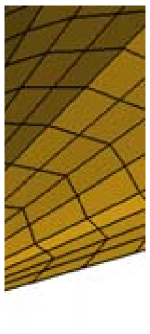
Part II: Local time-stepping

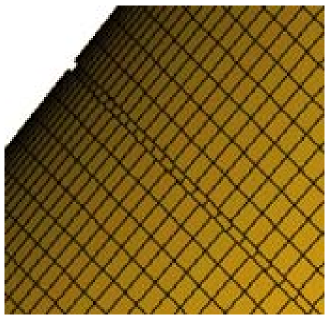
Problem: Small cells, even just one, cause a very small global time-step in an explicit scheme.

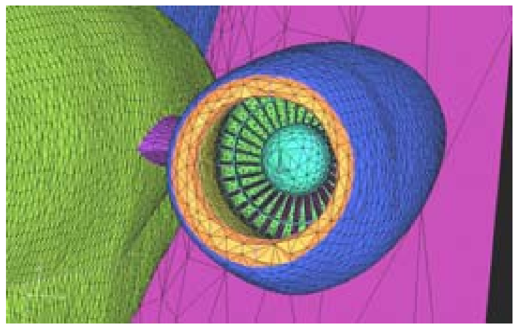


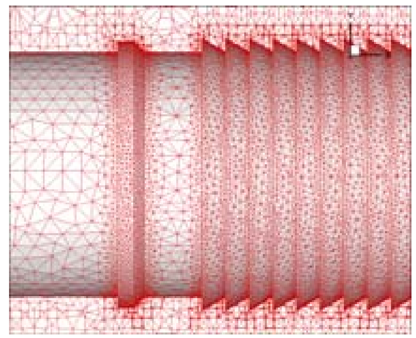
$$\Delta t \leq C\sqrt{\varepsilon\mu}\Delta x \simeq C_1\sqrt{\varepsilon\mu}\frac{N^2}{h}$$

A significant problem for large scale complex applications









<u>Old idea</u>: take only time-steps required by local restrictions.

<u>Old problems</u>: accuracy and stability

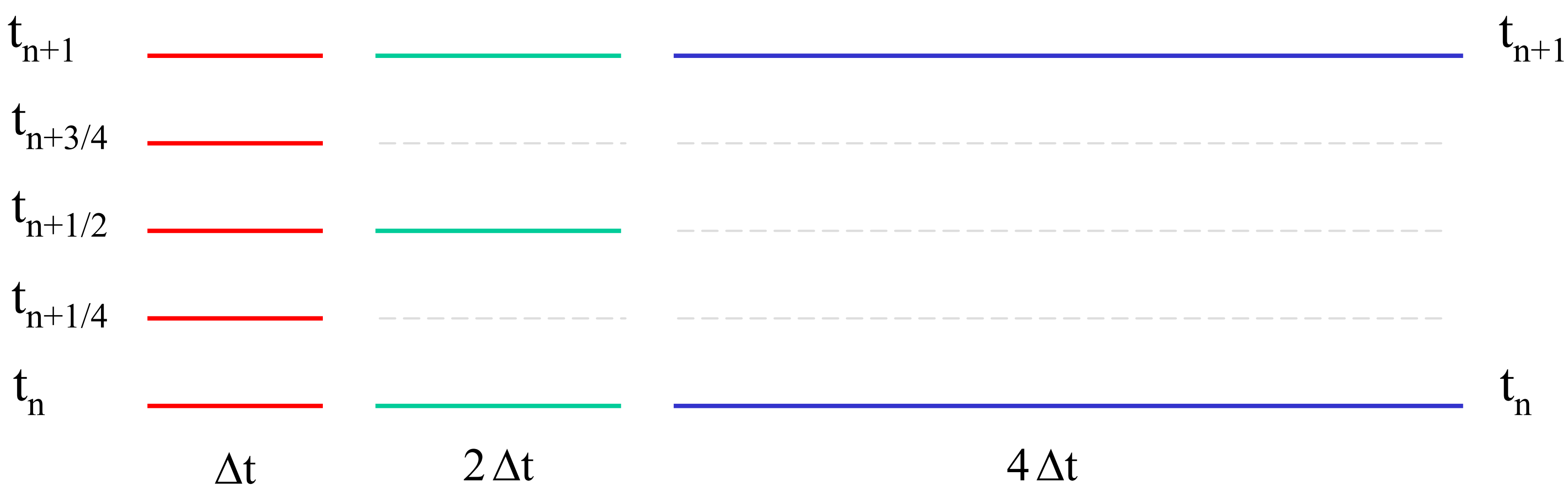
Local time-stepping

Substantial recent work by

Cohen, Grote, Lanteri, Piperno, Gassner, Munz etc

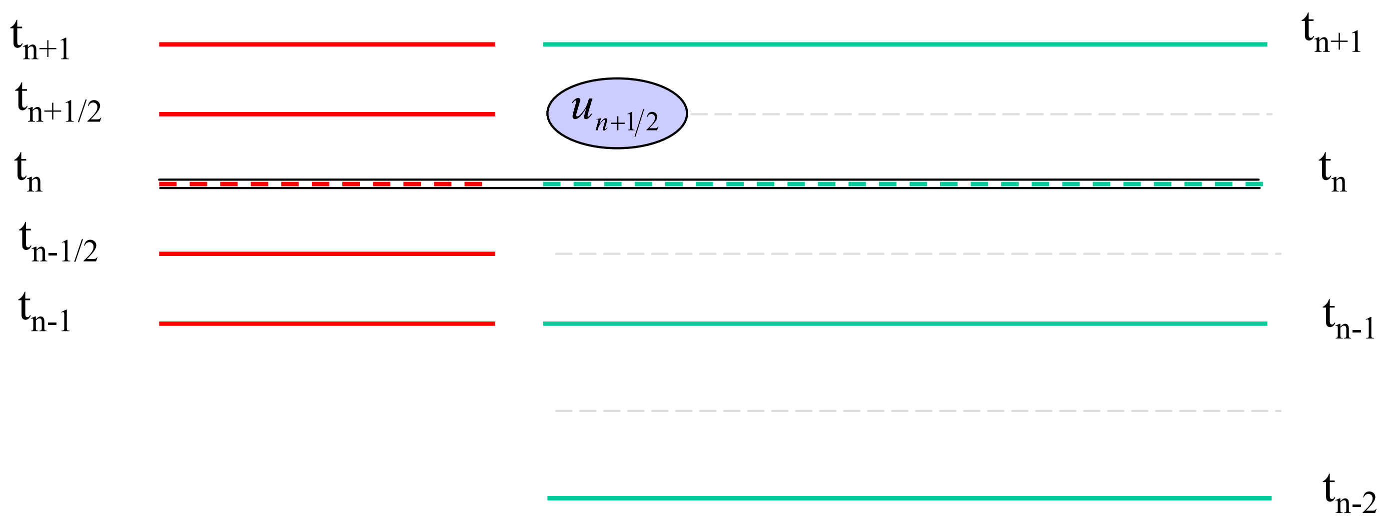
Most of the recent work is based on LF-like schemes, restricted to 2nd order in time.

Layout for **multi-rate** local time-stepping



Local time-stepping

Challenge: Achieving this at high-order accuracy

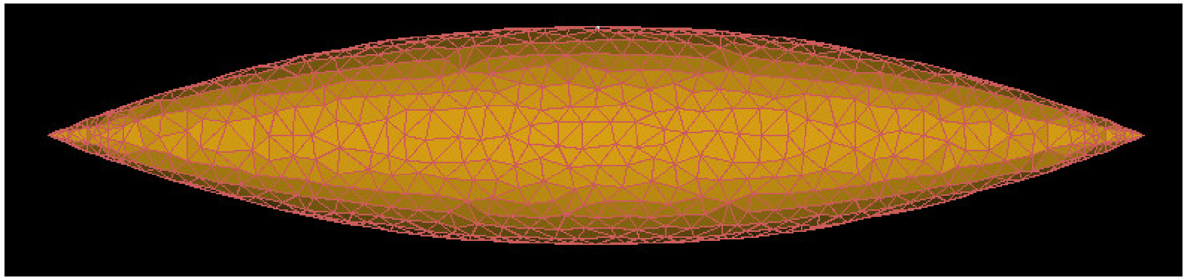


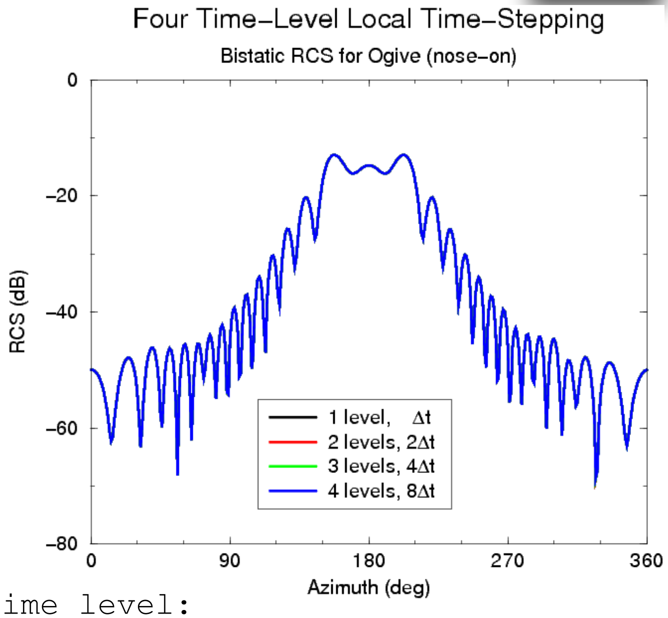
For all interior cells $$u_{n+1} = u_n + \frac{\Delta t}{12}\left[23F(u_n) - 16F(u_{n-1}) + 5F(u_{n-2})\right]$$

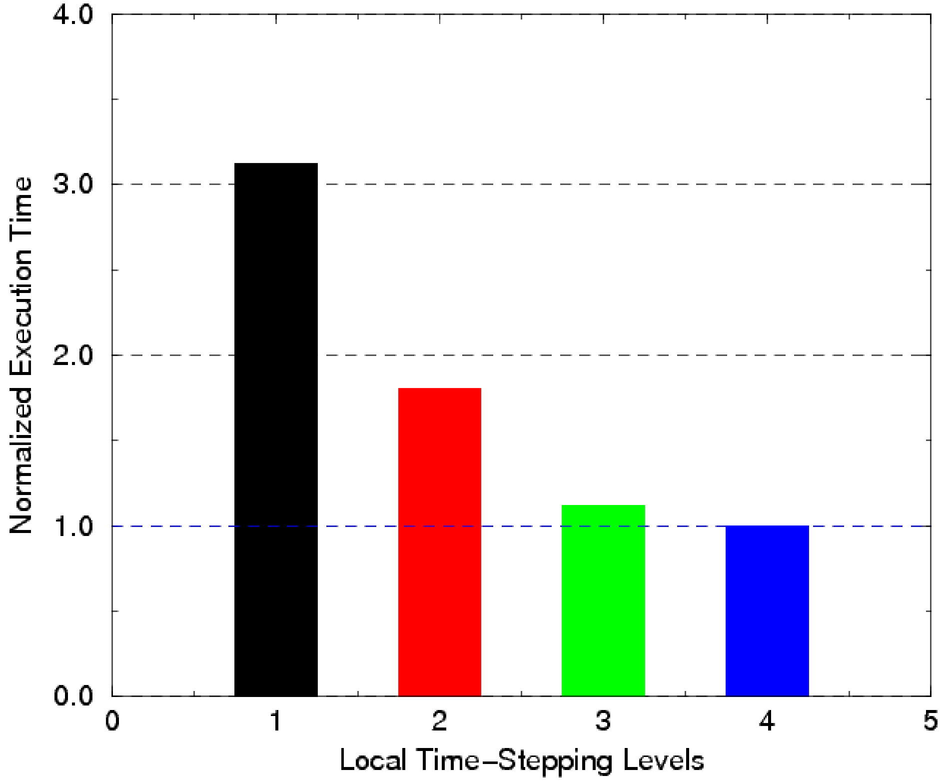
At interface cells $u_{n+1/2}$ $$u_{n+1/2} = u_n + \frac{\Delta t}{12}\left[17F(u_n) - 7F(u_{n-1}) + 2F(u_{n-2})\right]$$

This generalizes to many levels and arbitrary time-step fractions

Local time-stepping





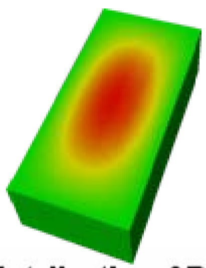


- One time level:
  - $N_o = 23742$
- Two time levels:
  - $N_o = 151$ (<1%)
  - $N_1 = 23591$ (99%)
- Three time levels:
  - $N_o = 151$ (<1%)
  - $N_1 = 1959$ (8%)
  - $N_2 = 21632$ (91%)
- Four time levels:
  - $N_o = 151$ (<1%)
  - $N_1 = 1959$ (8%)
  - $N_2 = 12622$ (53%)
  - $N_3 = 9010$ (38%)

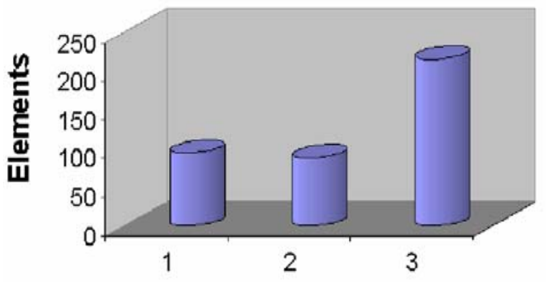
Computations by HyperComp Inc

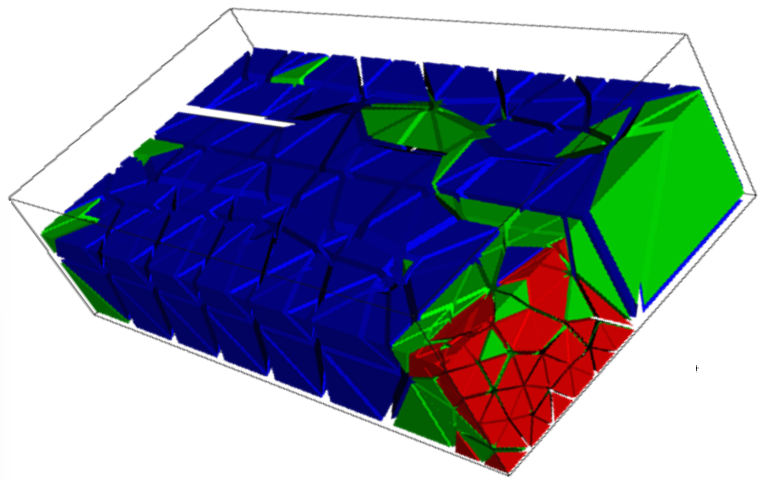
Local time-stepping

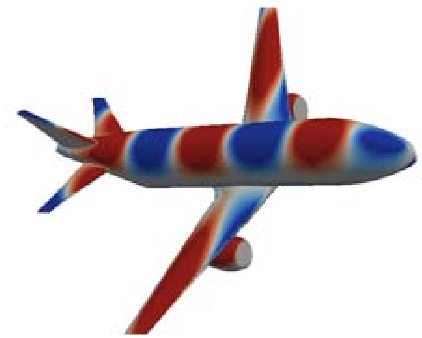
Segmentation is done in preprocessing



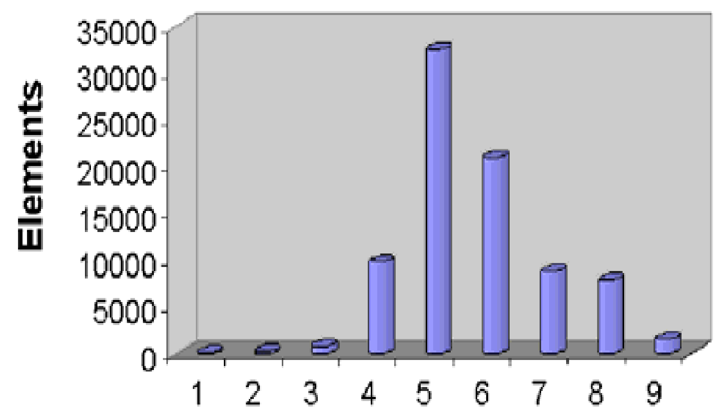
Level distribution 3D cavity







Level distribution airplane



Ideally suited for <u>local</u> DG scheme

<u>Known problems</u>:

No known stability proof

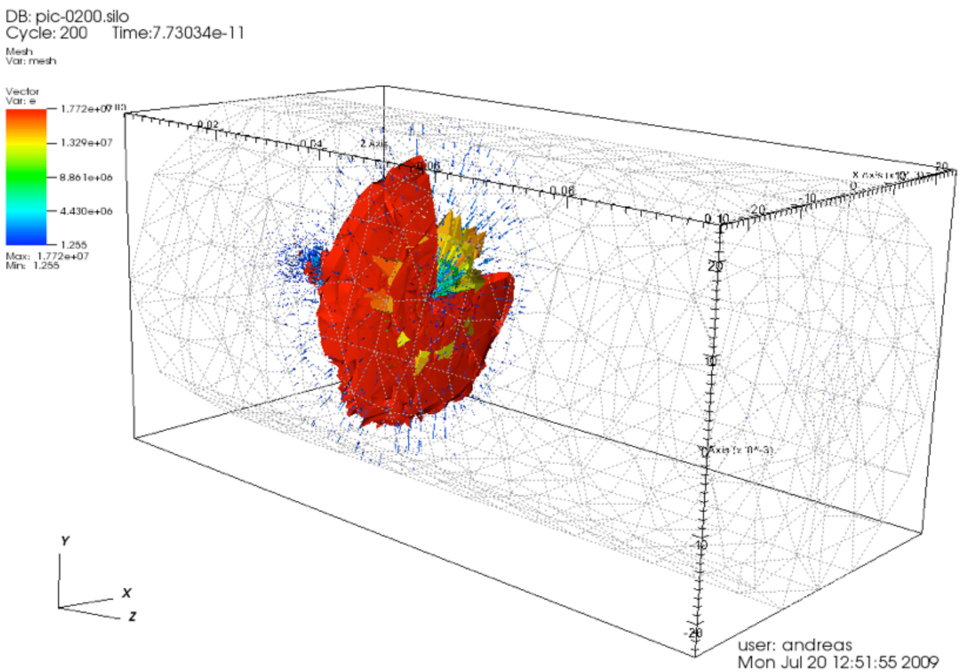
Time-step is not optimal (about 80%)

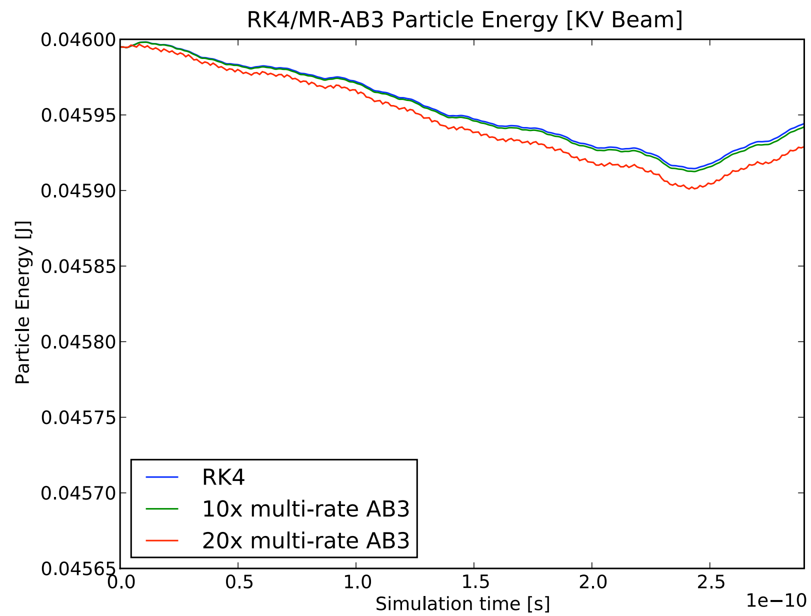
Extension to plasma physics/PIC

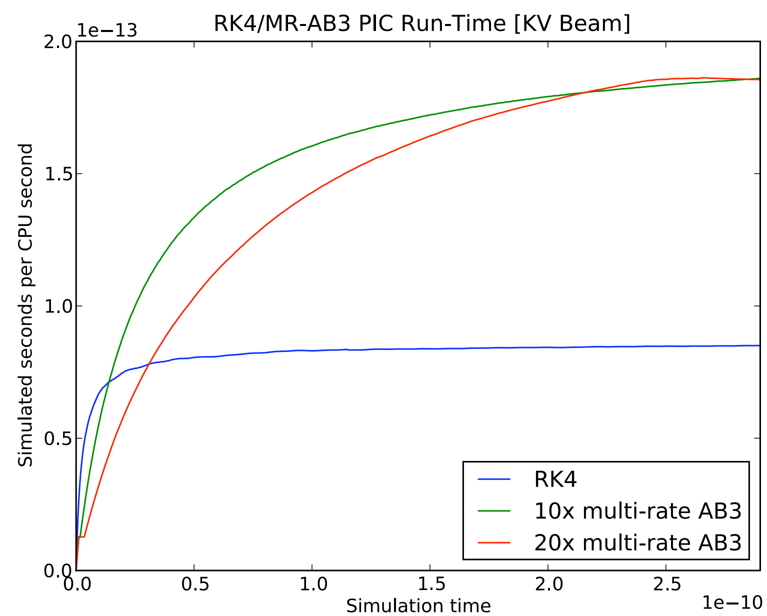
Basic approach -

✓ Do fields as fast scale

✓ Particles as slow scale







Extension to plasma physics/PIC

These are initial results

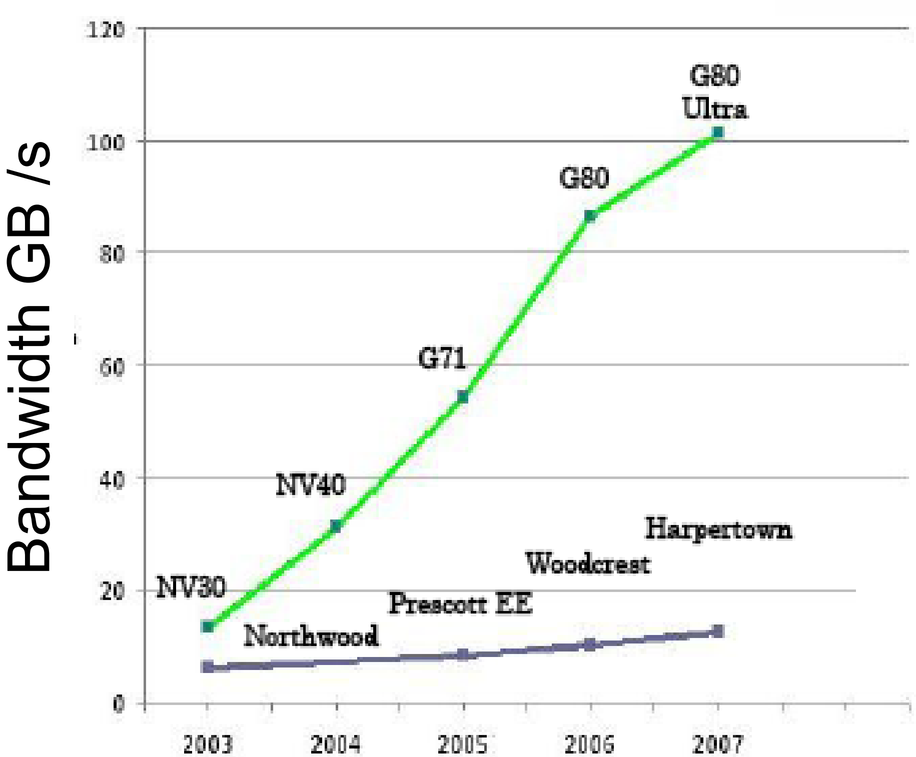
Significant potential for problems where :

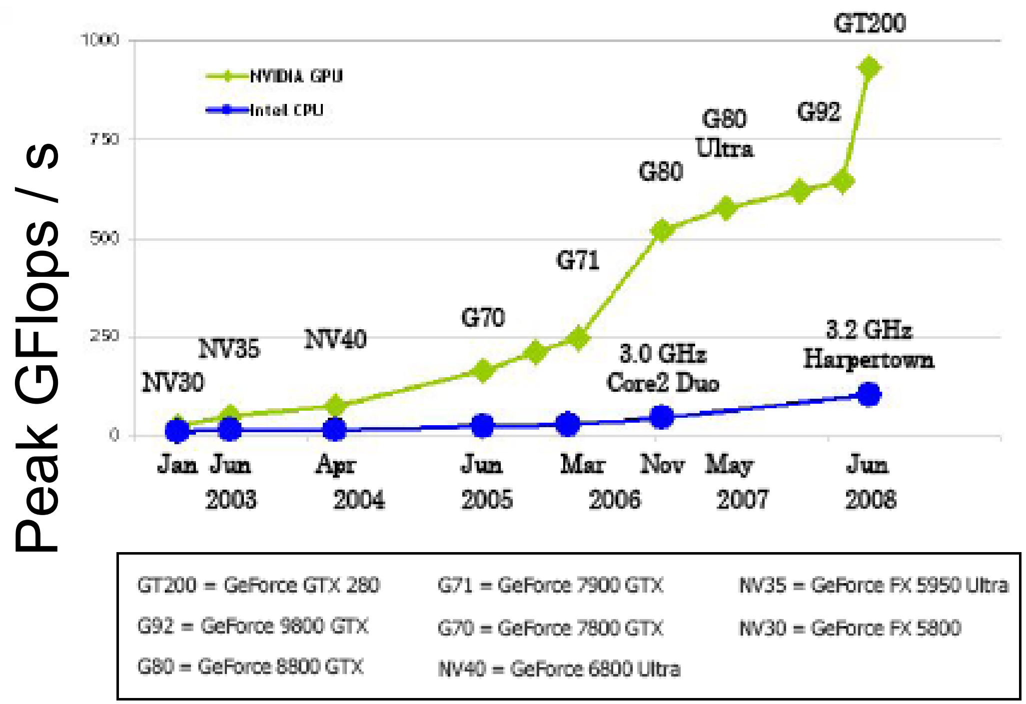
- ✓ Hyperbolic cleaning is used
- ✓ Significant grid induced stiffness
- ✓ Cost dominated by particle push

This is often the case for complex applications

Part III: CPUs vs GPUs

Notice the following





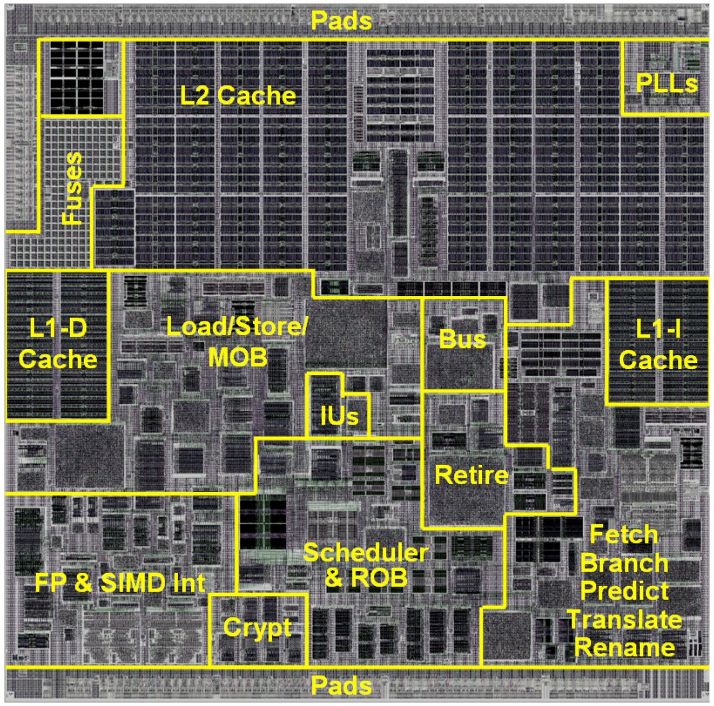
The memory bandwidth and the peak performance on Graphics cards (GPU's) is developing MUCH faster than on CPU's

At the same time, the mass-marked for gaming drives the prices down -- we have to find a way to exploit this !

But why is this ?

Target for CPU:

- ✓ Single thread very fast
- ✓ Large caches to hide latency
- ✓ Predict, speculate etc

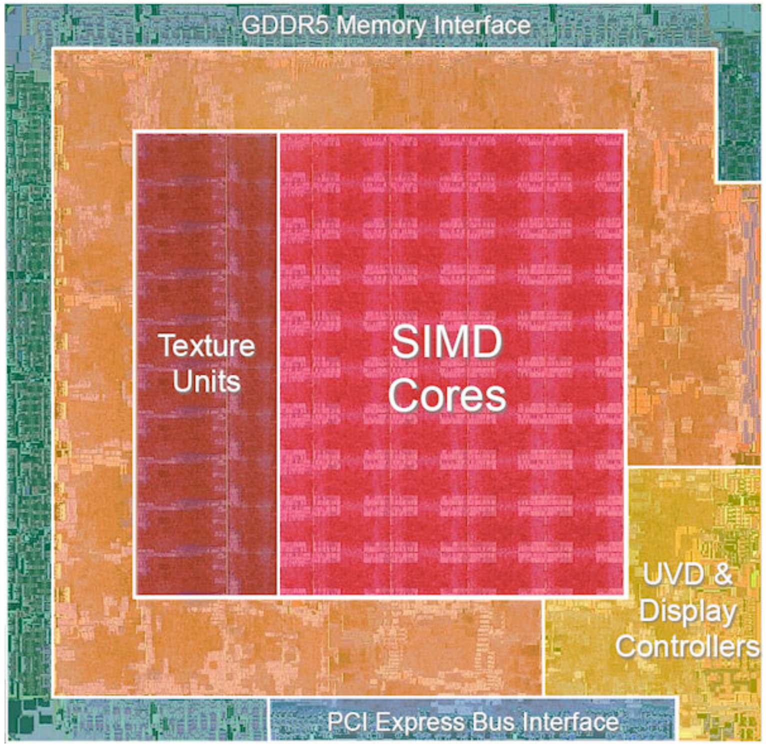


Lots of very complex logic to predict behavior

But why is this ?

For streaming/graphics cards it is very different

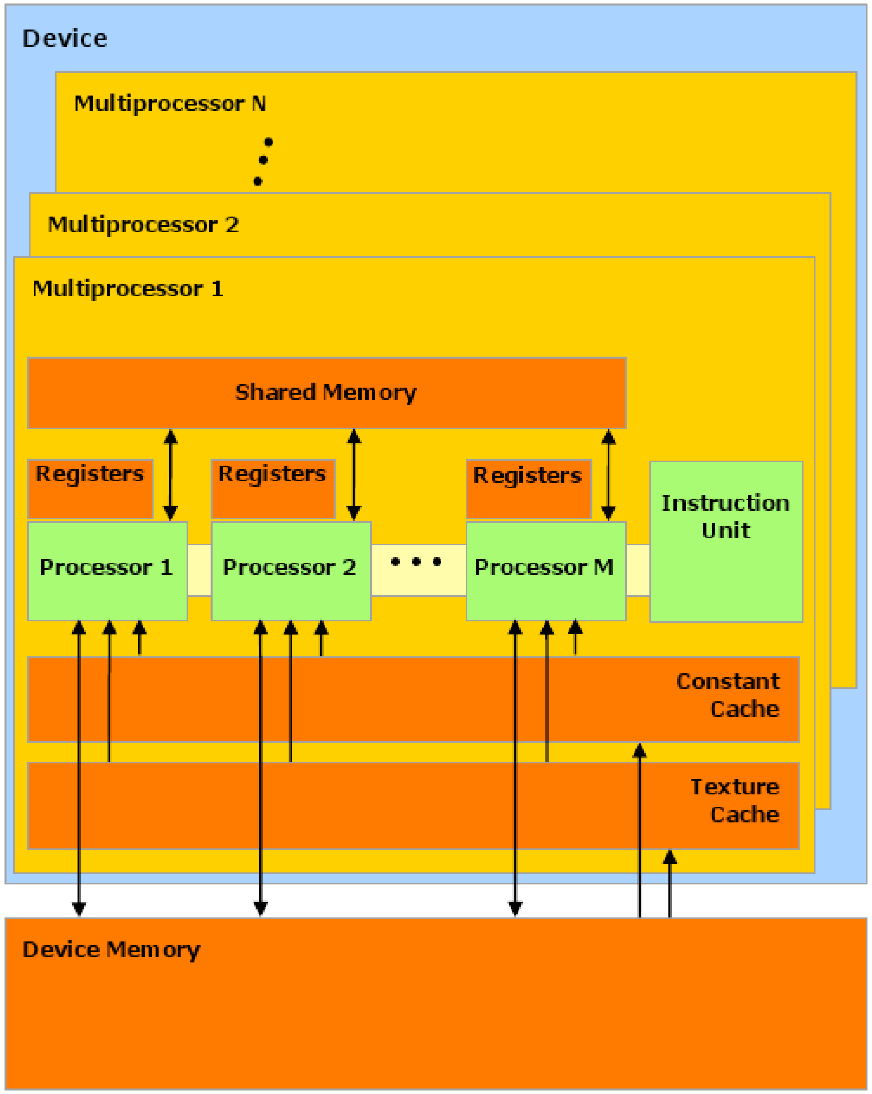
- ✓ Throughput is what matters
- ✓ Hide latency through parallelism
- ✓ Push hierarchy onto programmer



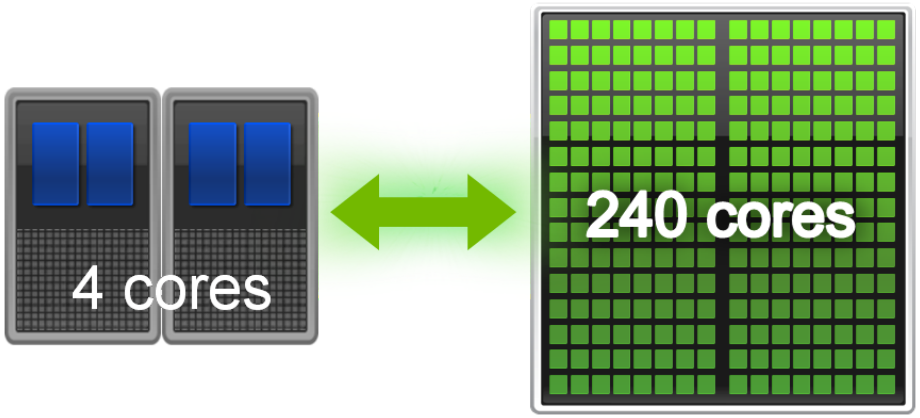
Much simpler logic with a focus on performance

GPUs 101

GPU layout



- ✓ 1 GPU = 30 MPs
- ✓ 1 MP has 1 IU, 8 SP, 1 DP
- ✓ 1 MP has 16KiB shared and 32 KiB Register memory
- ✓ 240 (512) threads
- ✓ Dedicated RAM at 140GB/s
- ✓ Limited caches

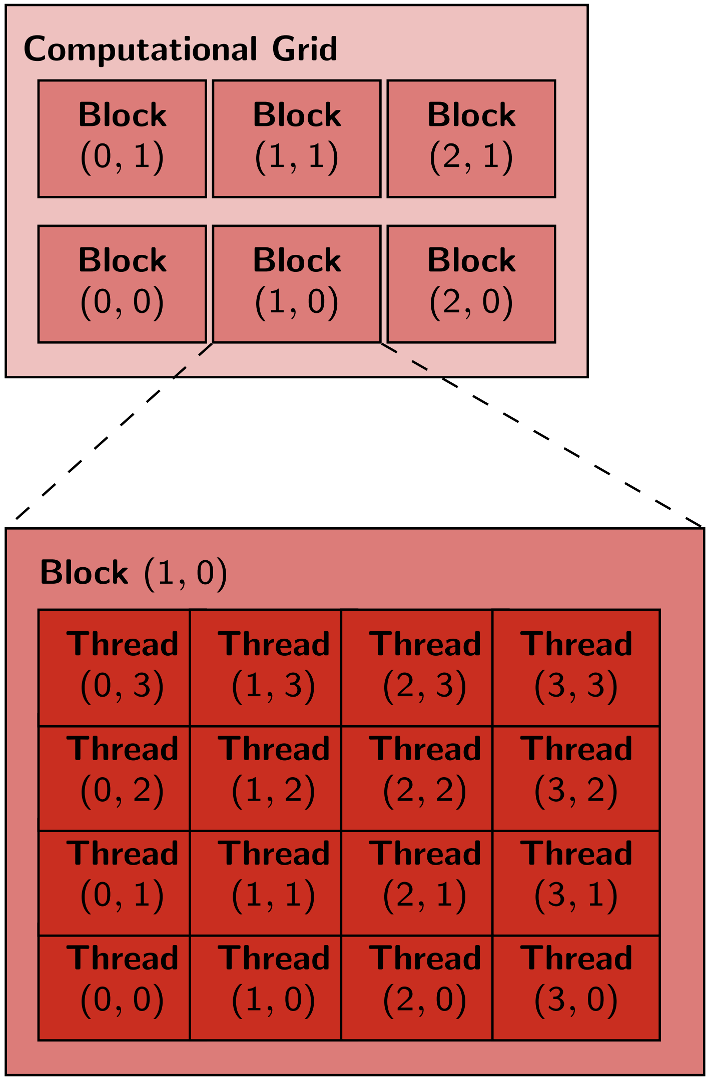


GPUs 101

Gains	Losses
⊕ Memory Bandwidth (140 GB/s vs. 12 GB/s)	⊖ Recursion
⊕ Compute Bandwidth (Peak: 1 TF/s vs. 50 GF/s, Real: 200 GF/s vs. 10 GF/s)	⊖ Function pointers
	⊖ Exceptions
	⊖ IEEE 754 FP compliance
	⊖ Cheap branches (i.e. `ifs`)

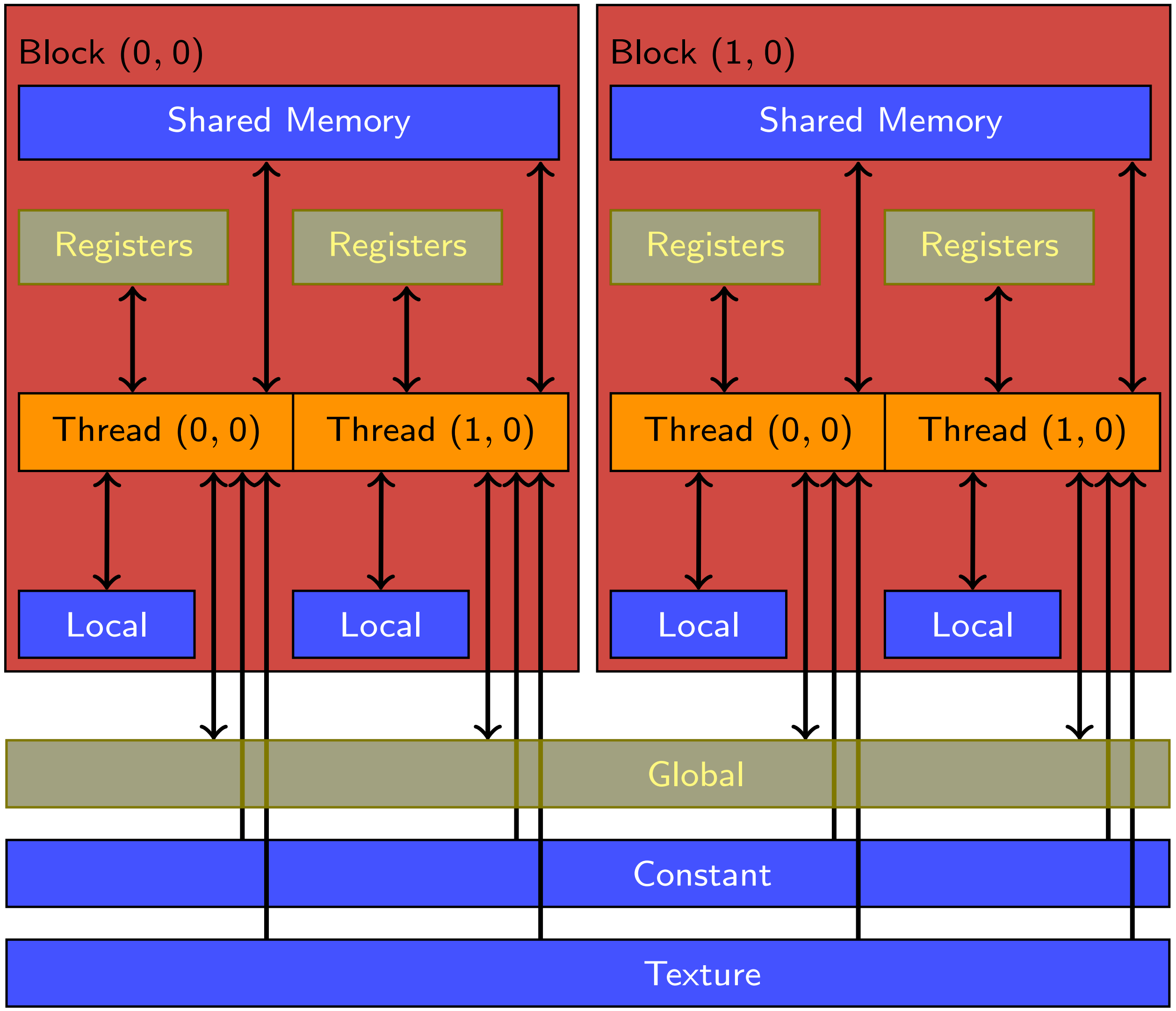
Already here it is clear that programming models/codes may have to undergo substantial changes -- and that not all will work well

GPUs 101



- ✓ Genuine multi-tiered parallelism
  - ✓ Grids
  - ✓ blocks
  - ✓ threads
- ✓ Only threads within a block can talk
  - ✓ Blocks must be executed in order
- ✓ Grids/blocks/threads replace loops
- ✓ Until recently, only single precision
- ✓ Code-able with CUDA (C-extension)

GPUs 101



Memory model:

- ✓ Registers
- ✓ Local shared
- ✓ Global

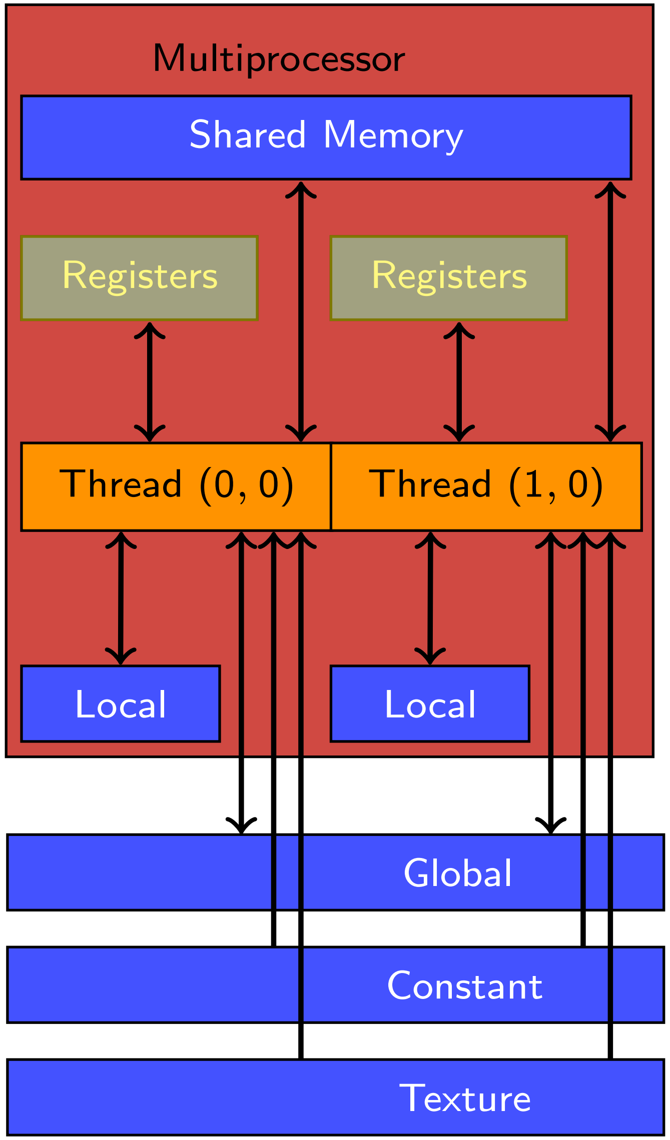
GPUs 101

✓ Lots of multi-processors (about 30)

... communicate through global mem

✓ Registers, shared memory, and threads communicate with low latency

... but memory is limited (16-32 KiB)



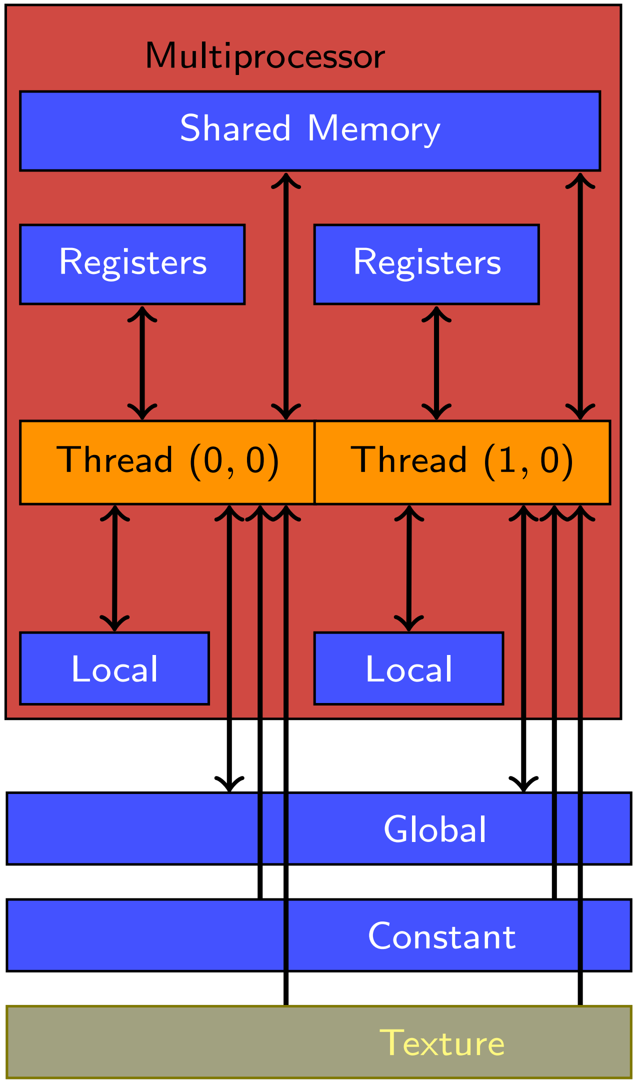
GPUs 101

✓ Global memory (4GiB/GPU) is plentiful

... but latency is high (512 bit bus)
... and stride one is preferred

✓ Texture is similar to global memory

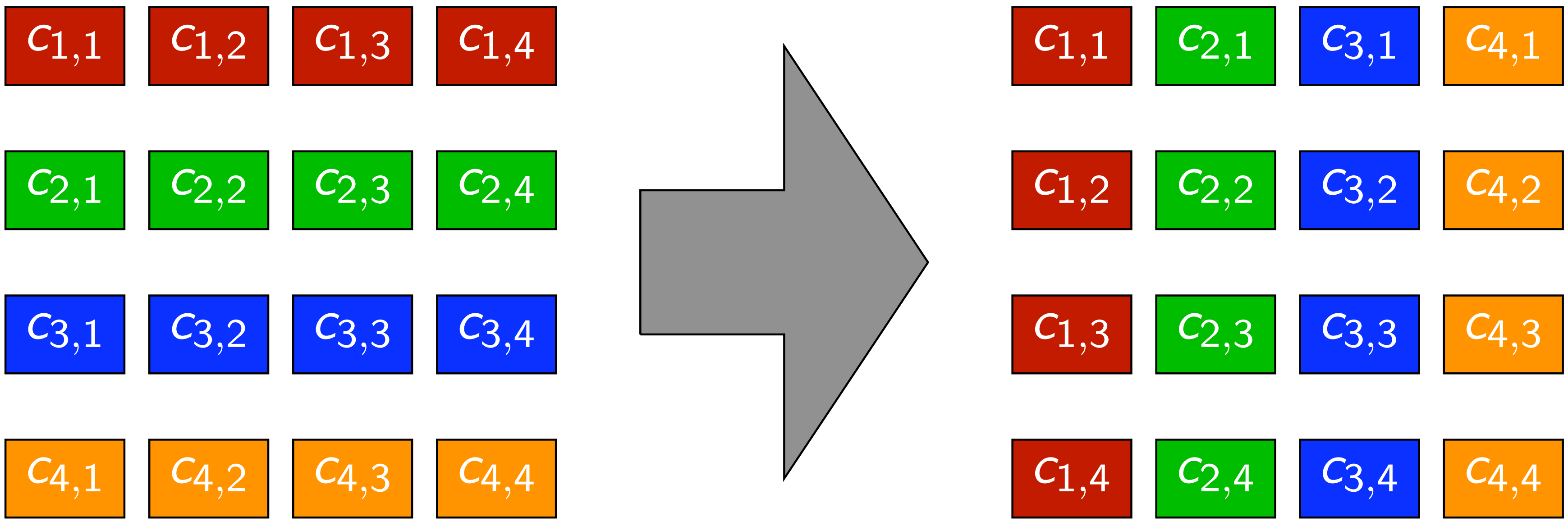
... allows more general access patterns
... but it is read only



Type	Per	Access	Latency
Registers	thread	R/W	1
Local	thread	R/W	1000
Shared	block	R/W	2
Global	grid	R/W	1000
Constant	grid	R/O	1-1000
Texture	grid	R/O	1000

Let’s consider an example

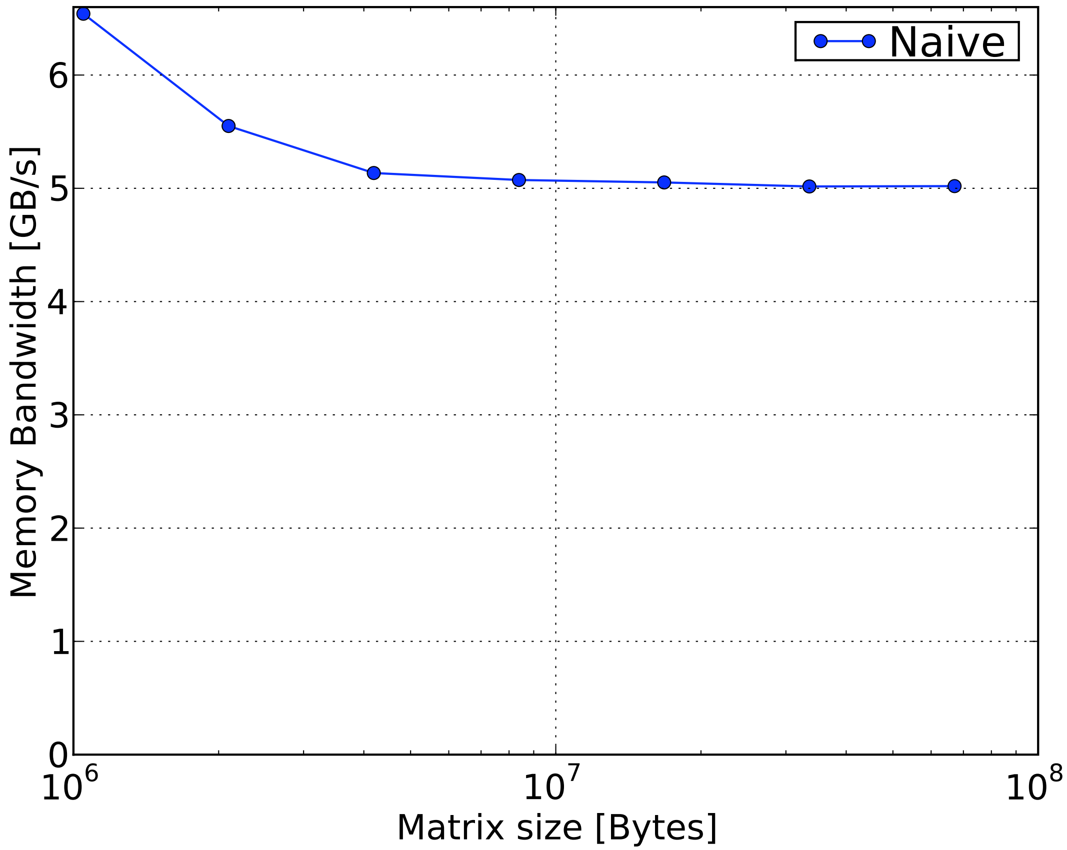
Matrix transpose



Memory bandwidth will be a limit here

Let's consider an example

Using just global memory



As CPU

Reading from global mem:

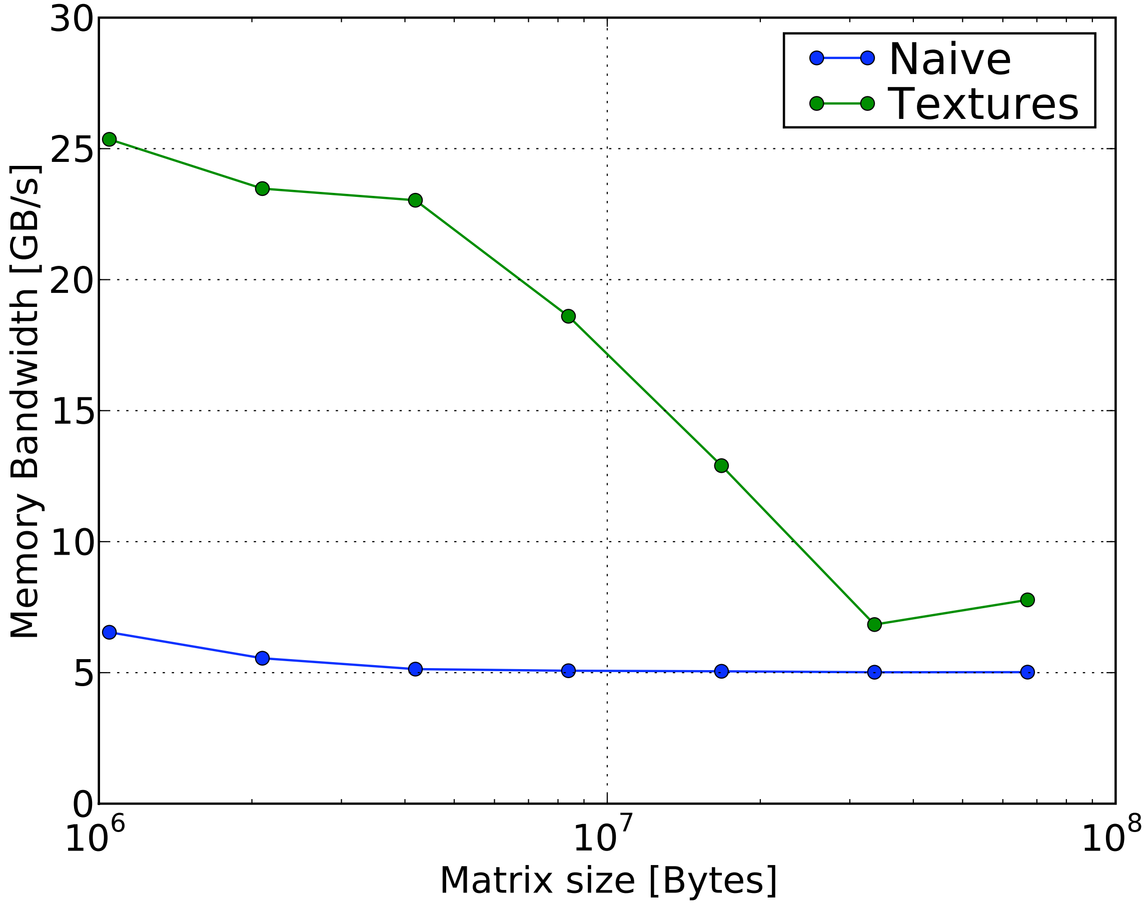
Writing to global mem:

stride: 1 $\longrightarrow$ one mem.trans.

stride: 16 $\longrightarrow$ **16 mem.trans.!**

Let's consider an example

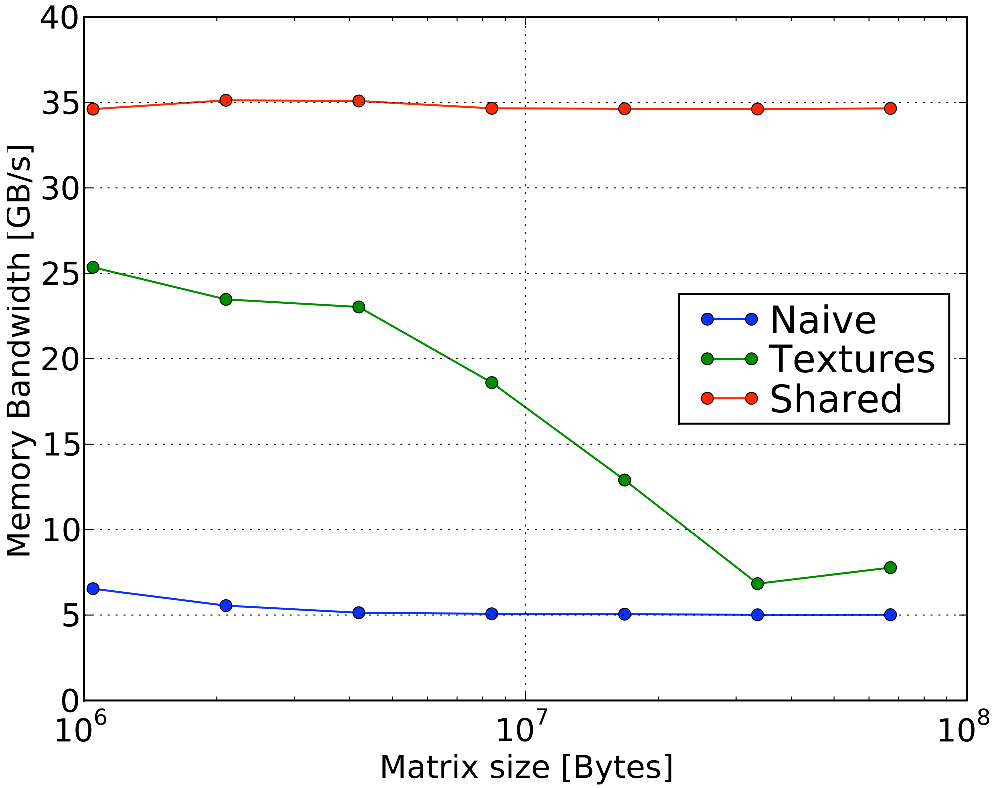
Using just texture(read)+global(write) memory



Getting better

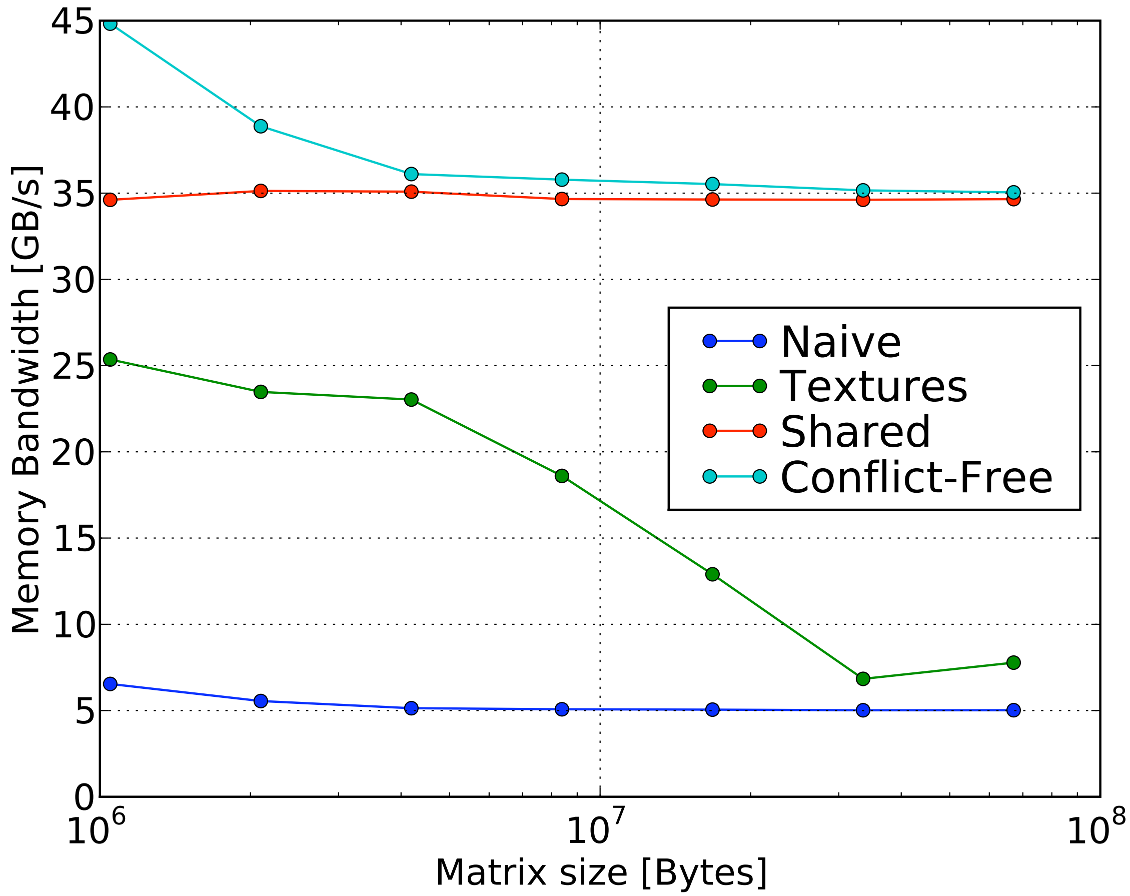
Let's consider an example

Transpose block-by-block in shared memory - this does not care about strides



Let's consider an example

Additional improvements are possible for small matrices - bank conflicts in shared memory

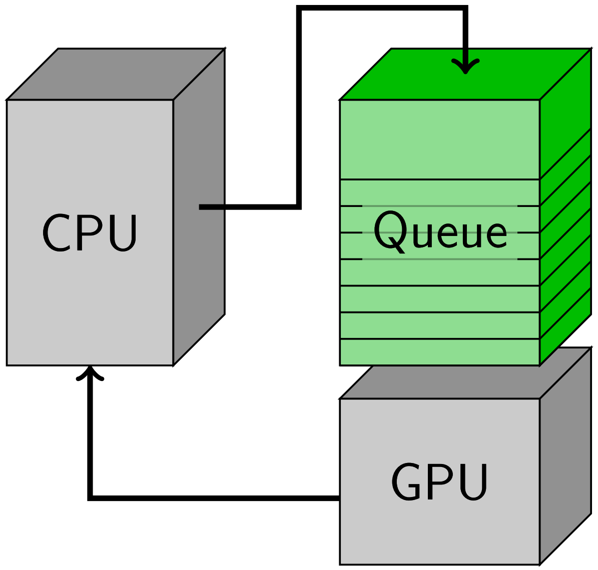


<u>A factor of 7-8 over CPU</u>

CPUs vs GPUs

The CPU is mainly the traffic controller
... although it need not be

- ✓ The CPU and GPU runs asynchronously
- ✓ CPU submits to GPU queue
- ✓ CPU synchronizes GPUs
- ✓ Explicitly controlled concurrency is possible



GPUs overview

- ✓ GPUs exploit multi-layer concurrency
- ✓ The memory hierarchy is deep
- ✓ Memory padding is often needed to get optimal performance
- ✓ Several types of memory must be used for performance

- ✓ First factor of 5 is not too hard to get
- ✓ Next factor of 5 requires quite some work
- ✓ Additional factor of 2-3 requires serious work

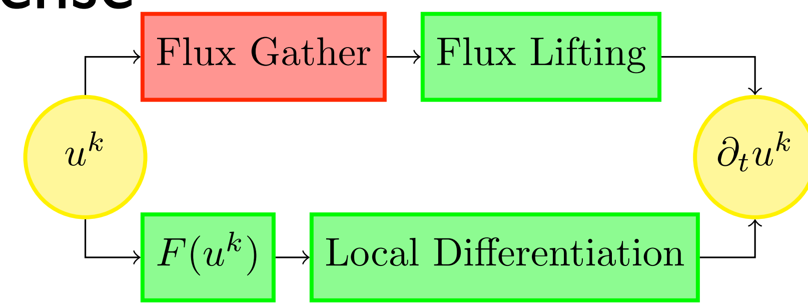
Nodal DG on GPU's

So what does all this mean ?

✓ GPU's has deep memory hierarchies so local is good
  ➡ The majority of DG operations are local

✓ Compute bandwidth >> memory bandwidth
  ➡ High-order DG is arithmetically intense

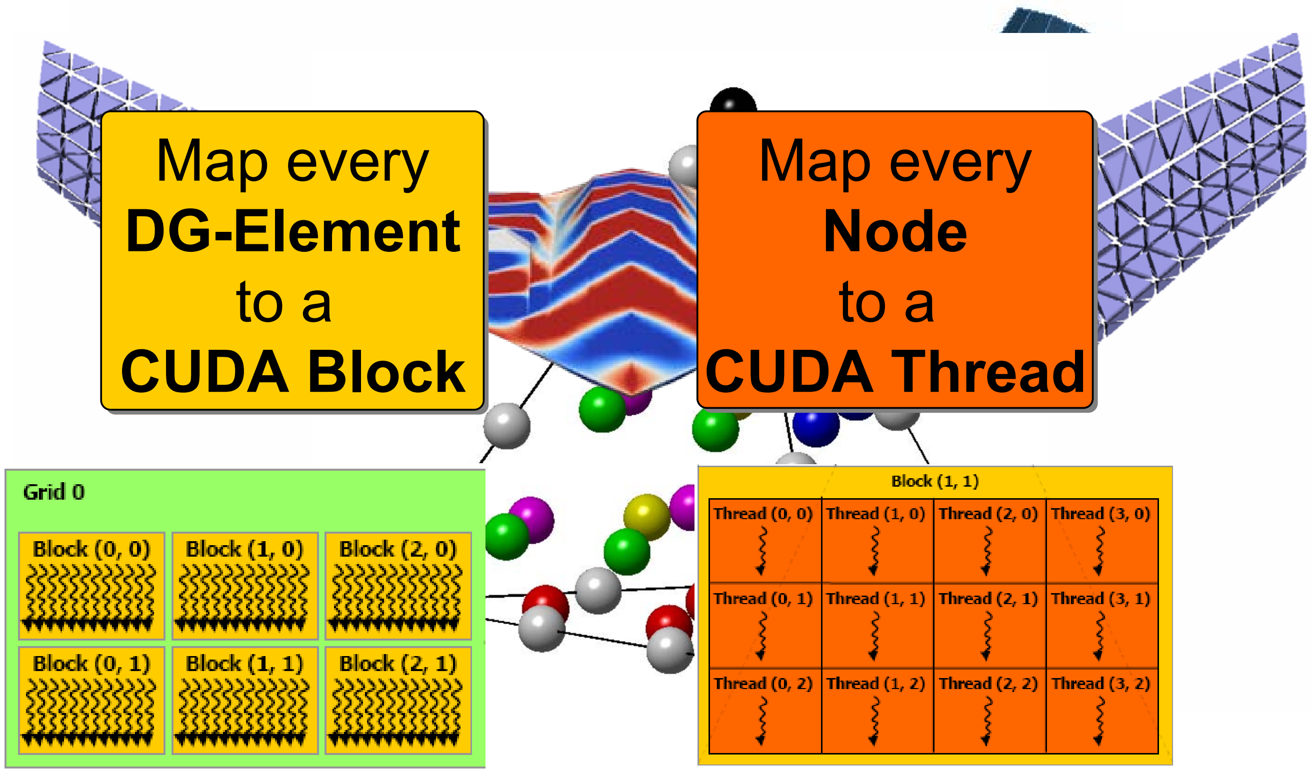
✓ GPU global memory favors dense data
  ➡ Local DG operators are all dense



With proper care we should be able to obtain excellent performance for DG-FEM on GPU's

Nodal DG on GPU's

Nodes in threads, elements in blocks

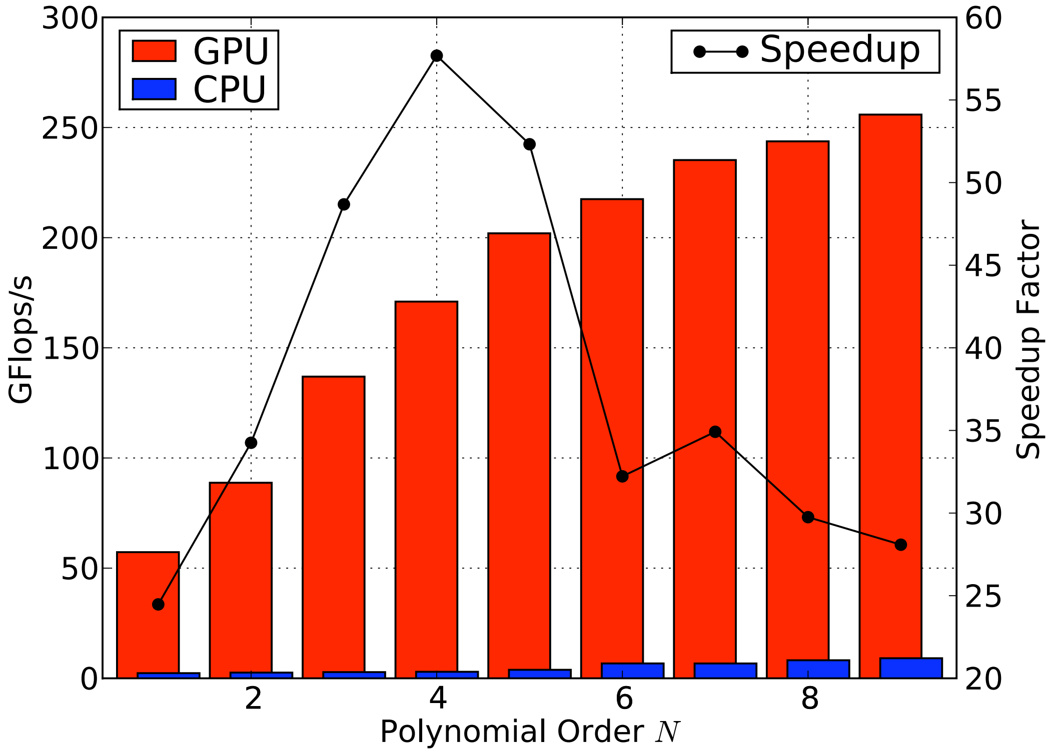


Other choices:

✓ D-matrix in shared, data in global (small N)

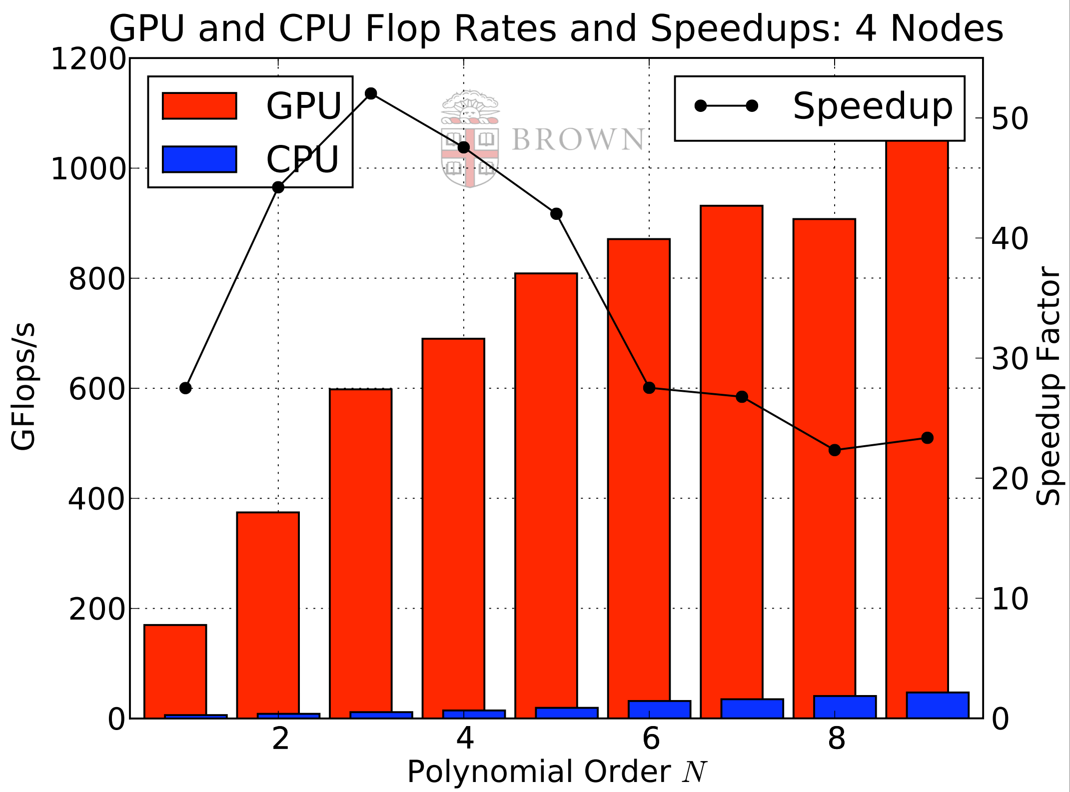
✓ Data in shared, D-matrix is global (large N)

Nodal DG on GPU's



DG-FEM on one GPU

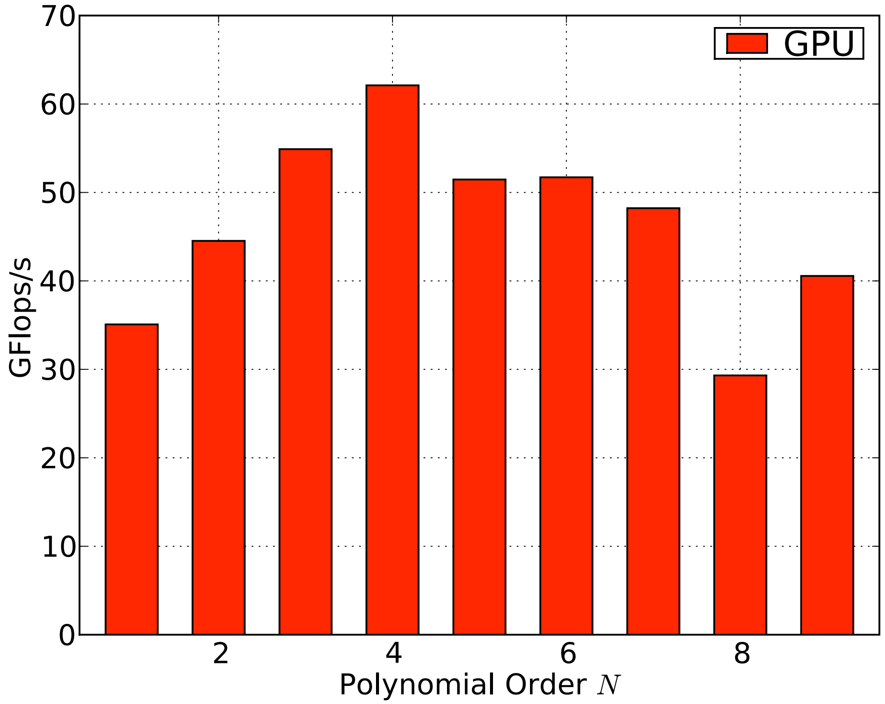
DG-FEM on four GPU
one card

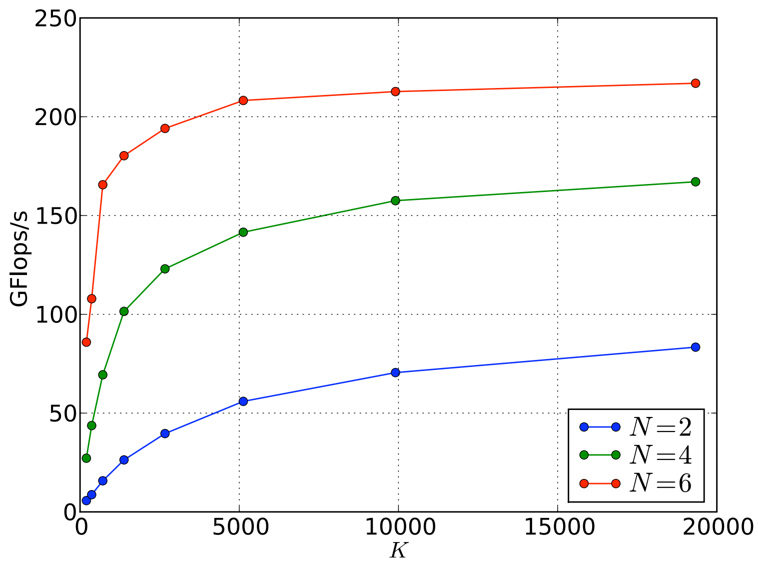


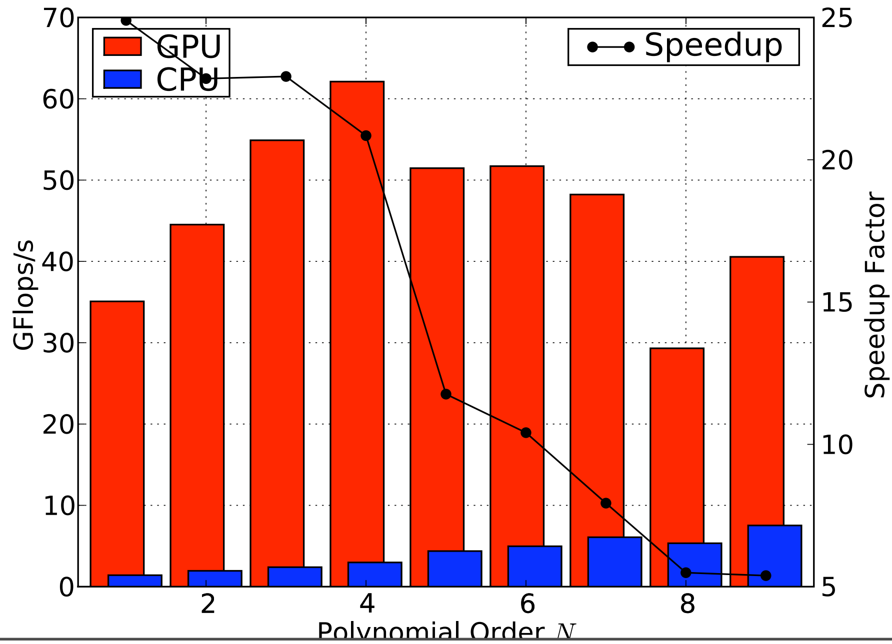
Nodal DG on GPU's

Where you need it most

... and for larger and larger grids

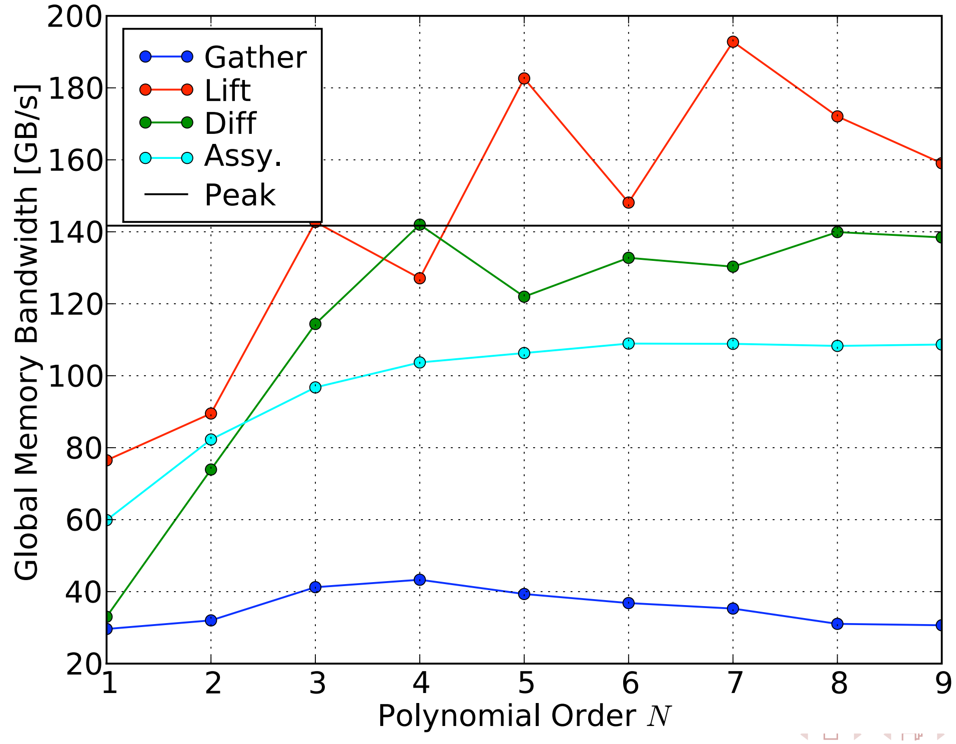






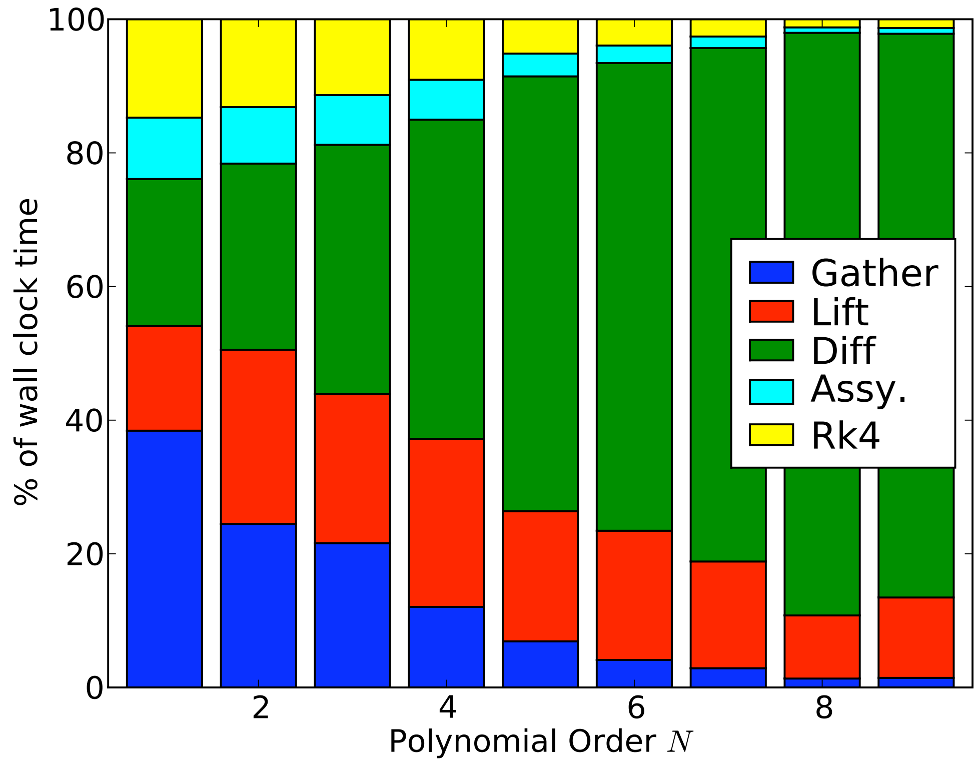
Also in double precision

Nodal DG on GPU's



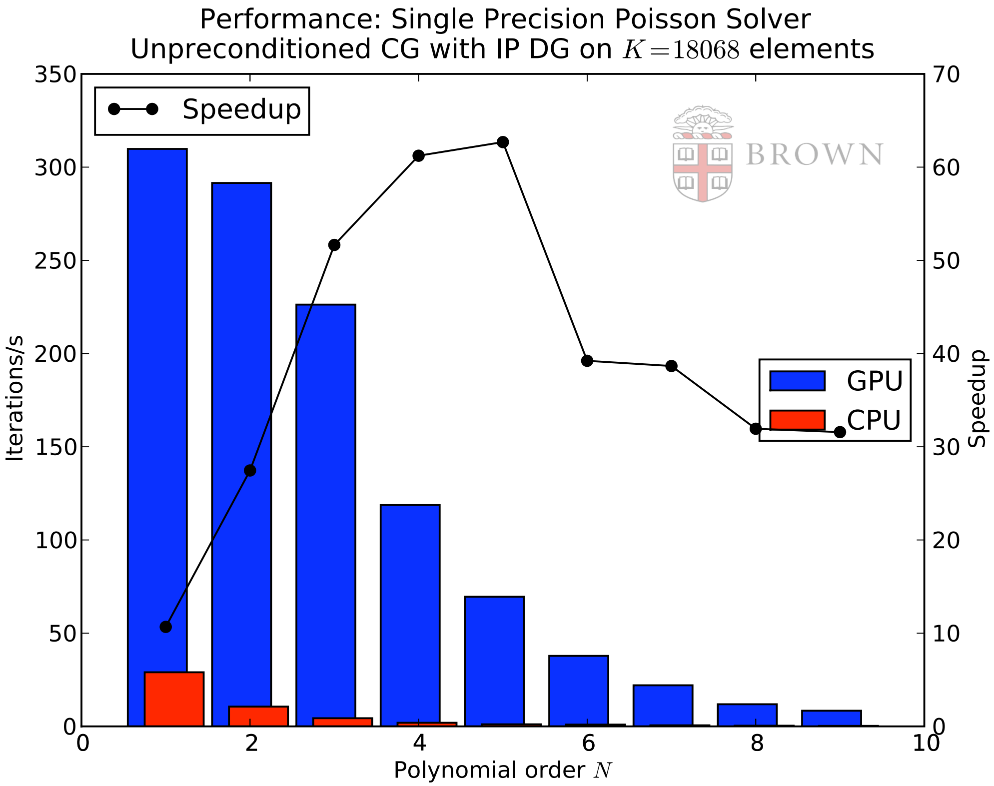
Efficient utilization of memory bandwidth

Utilization of resources where they matter most

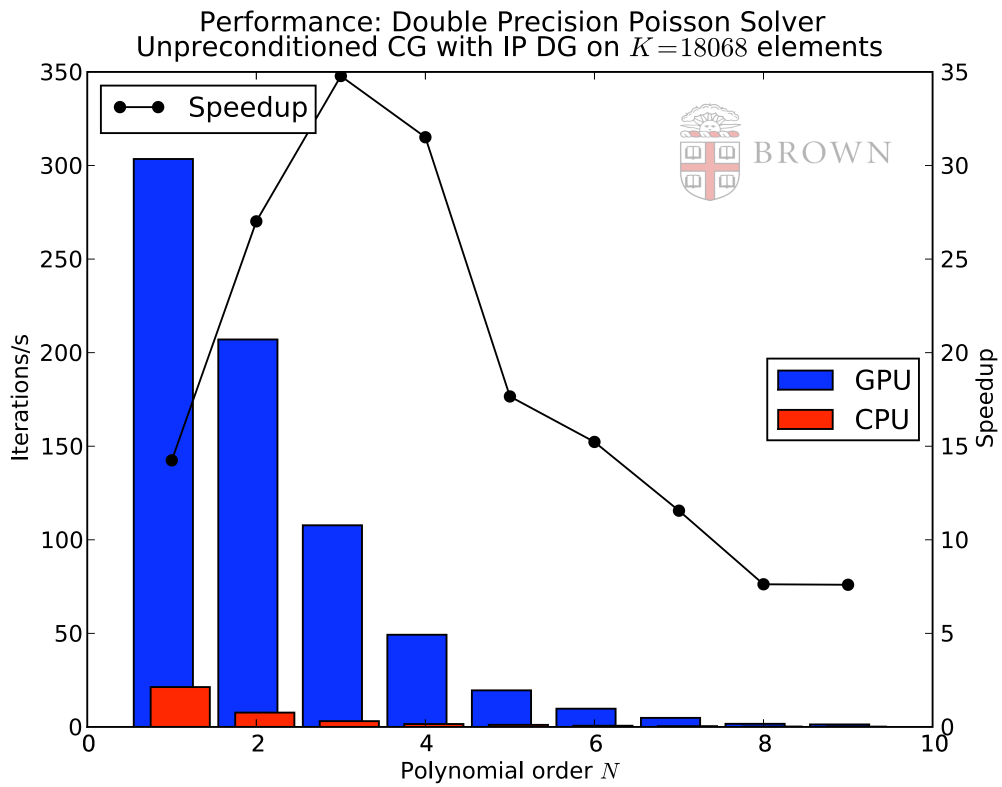


Nodal DG on GPU's

Similar results for DG-FEM Poisson solver with CG

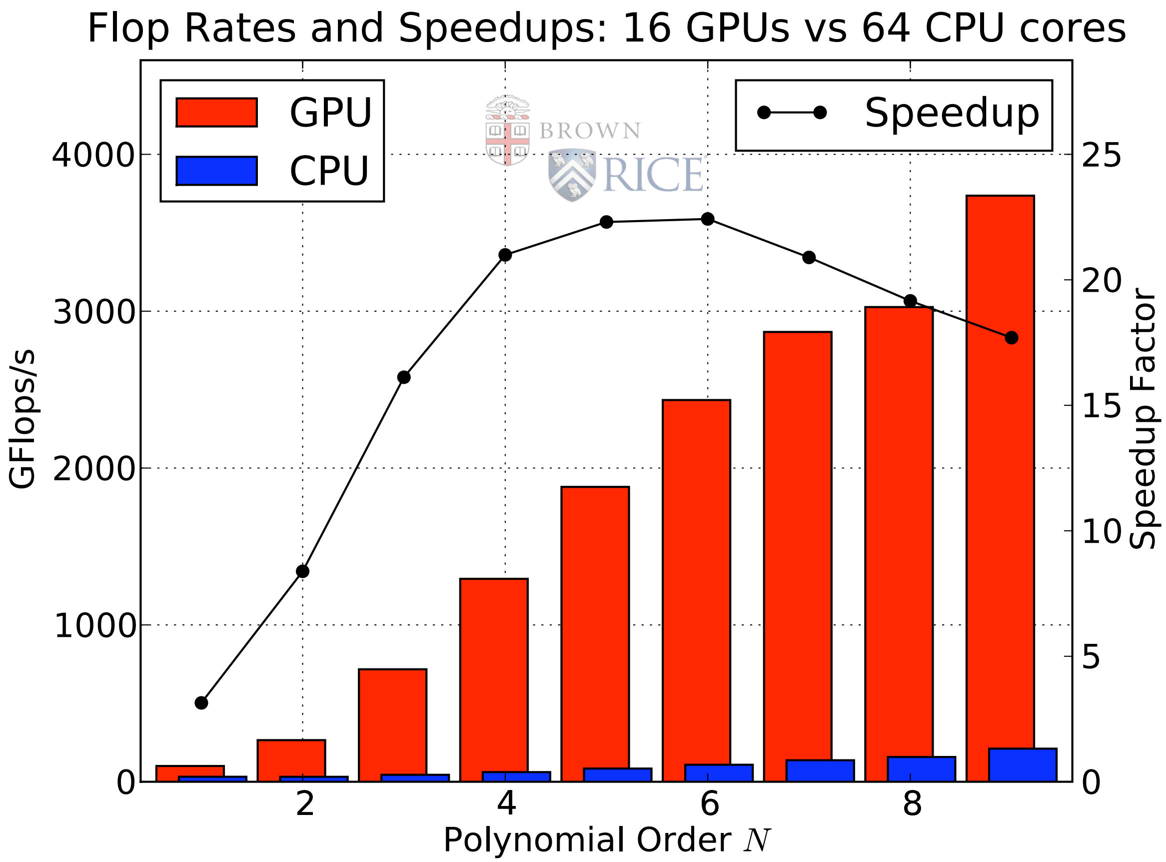


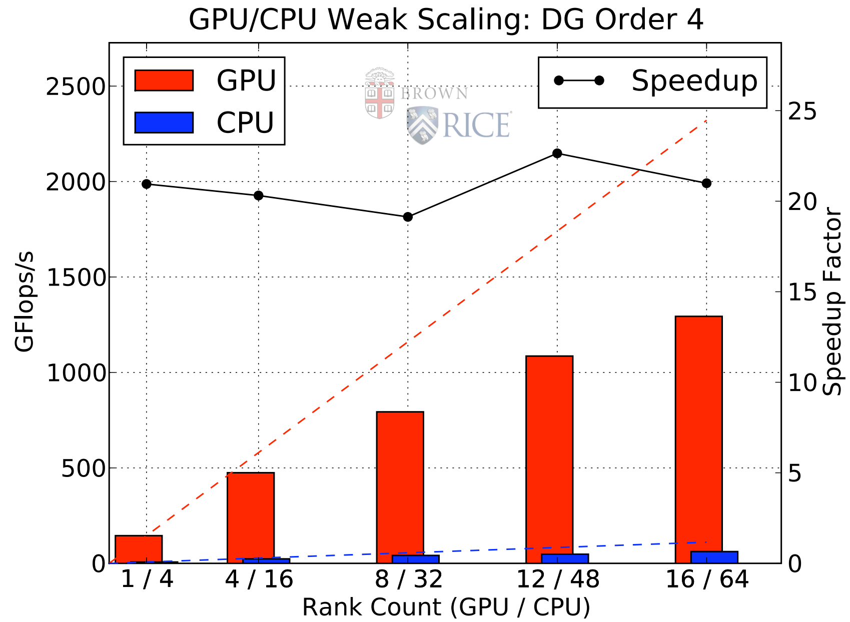
Note: No preconditioning

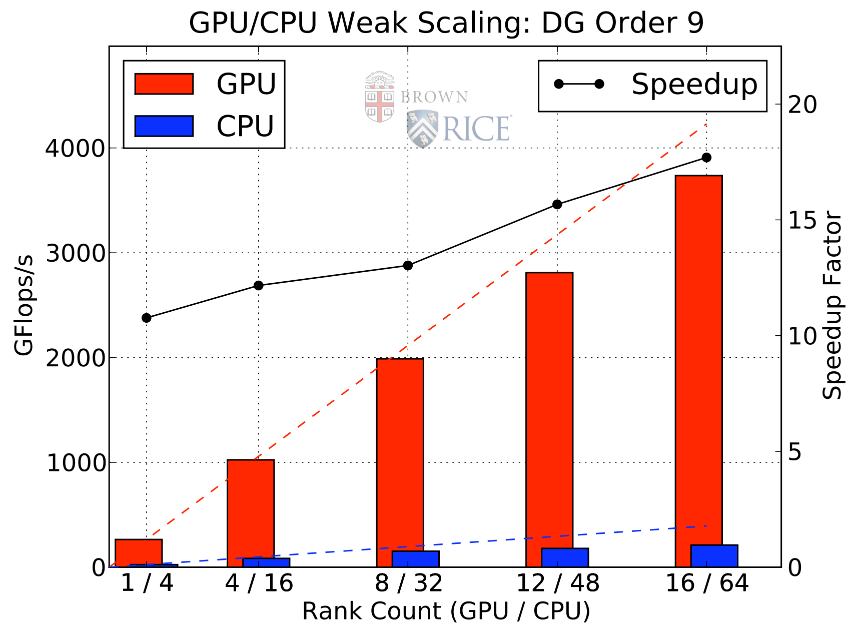


Combined GPU/MPI solution

MPI across network

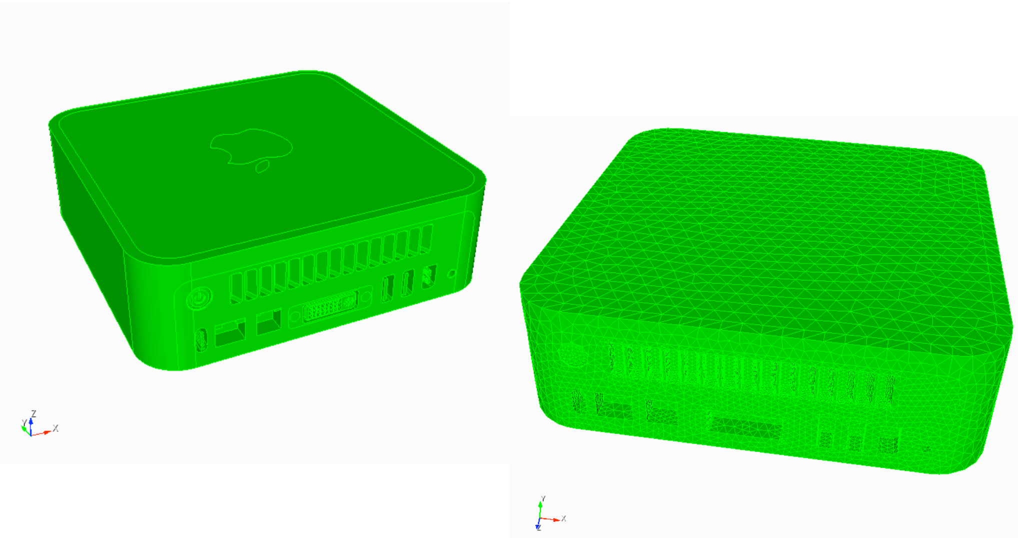


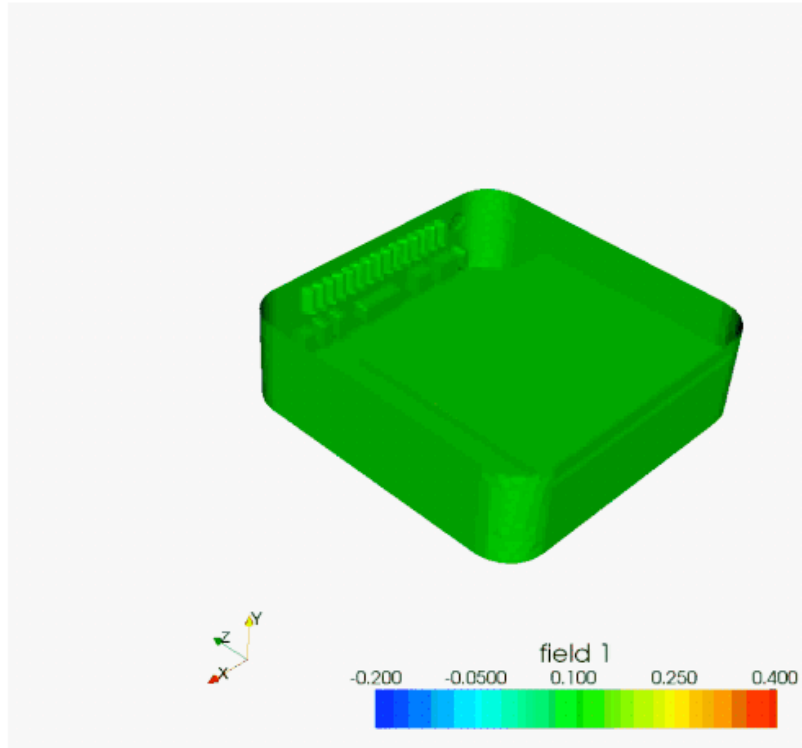


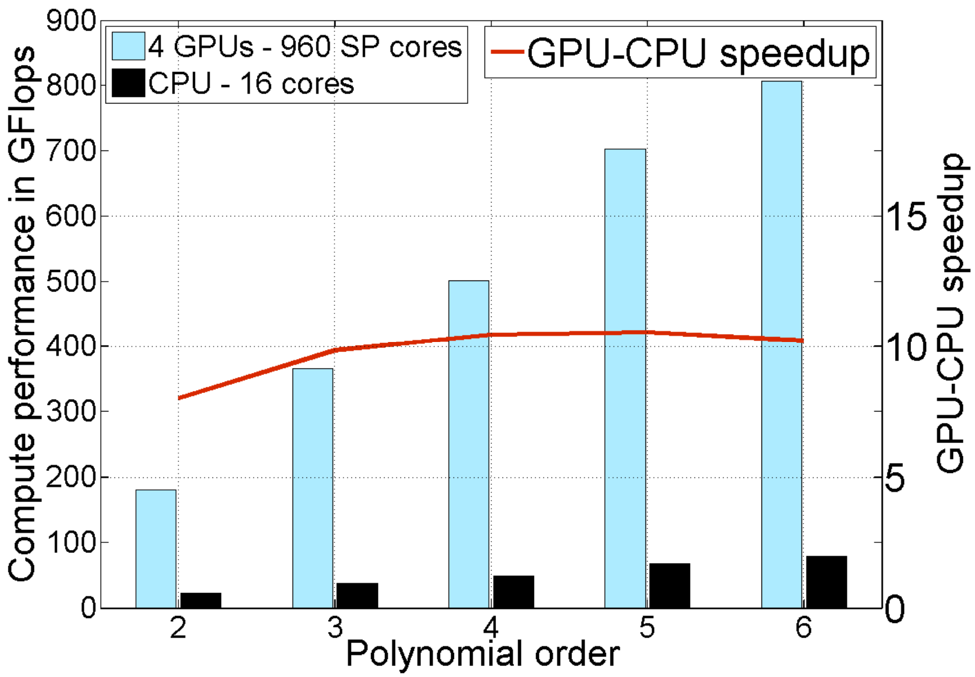


Good scaling when problem is large

Example - a Mac Mini



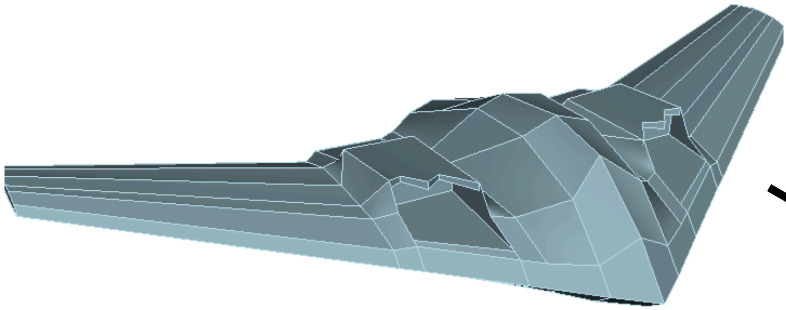


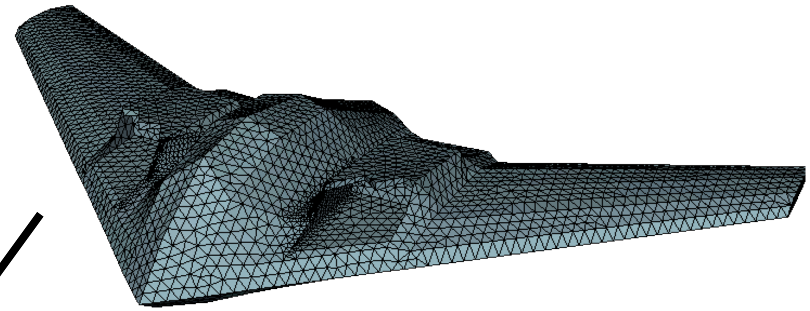


K=201765 elements
3rd order elements

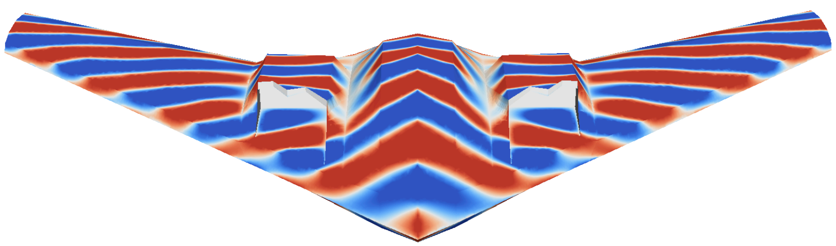
Computation by N. Godel

Example: Military aircraft

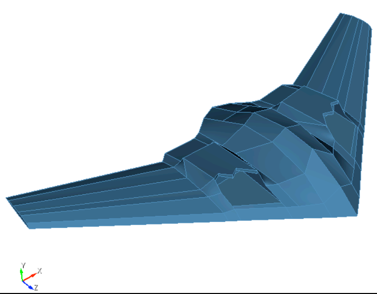




K=130413 elements
3rd order elements
15.6E6 DoF



Computation by N. Godel

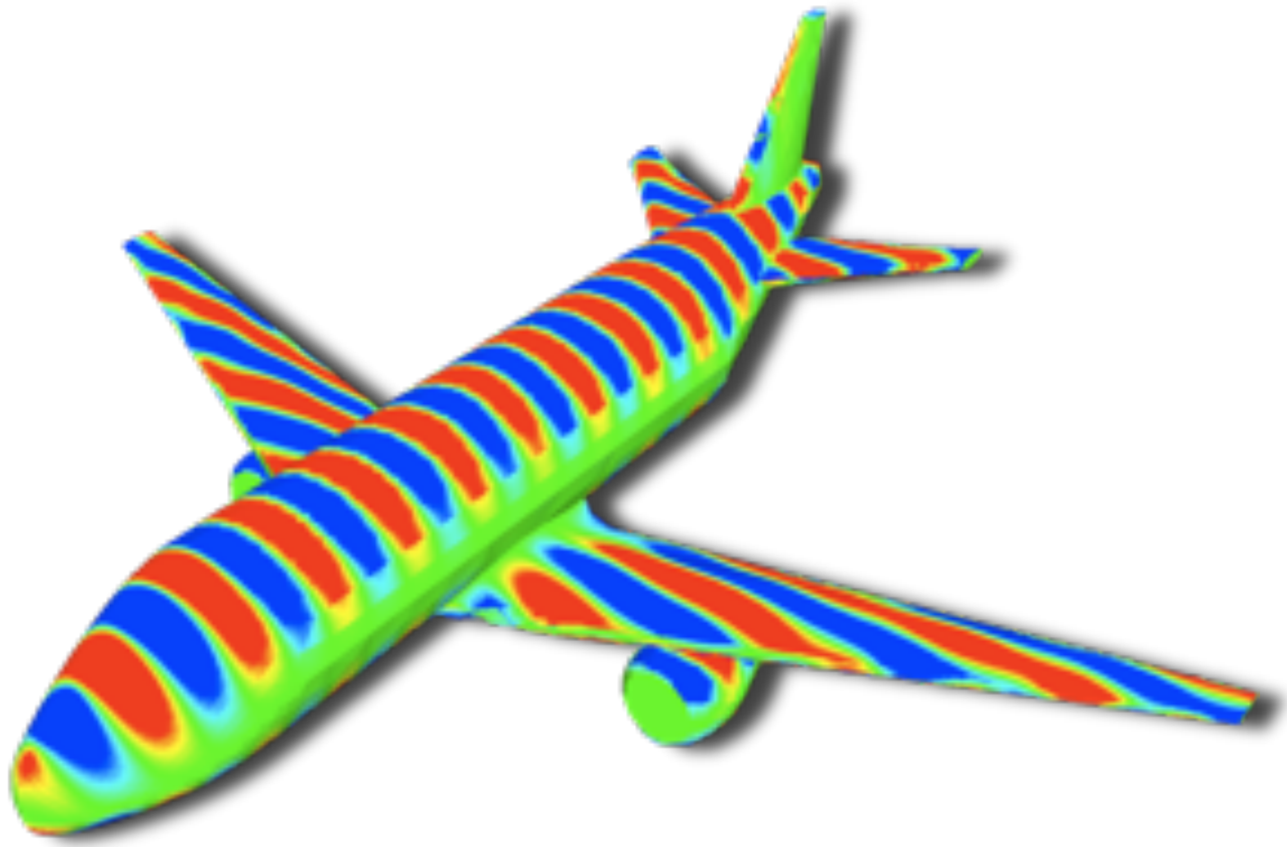
	CPU global	29 h 6 min 46 s	1.0
	GPU global	39 min 1 s	44.8
	GPU multirate	11 min 50 s	147.6

Nodal DG on GPU's

Not just for toy problems

228K elements
5th order elements
78m DOF
68k time-steps

Time ~ 6 hours



711.9 GFlop/s on one card

Computation by N. Godel

Nodal DG on GPU's

Several GPU cards can be coupled over MPI at minimal overhead (demonstrated). Lets do the numbers

One 700GFlop/s/4GB mem card costs ~$8k

So $250k will buy you 16-18TFlop/s sustained

This is the entry into Top500 Supercomputer list !

... at 5%-10% of a CPU based machine

This is **a game changer** -- and the local nature of DG-FEM makes it very well suited to take advantage of this

Concluding remarks

While high order methods in general and DG-FEM in particular are widely used, there are still things to be done.

Combining

✓ New non-polynomial basis functions
✓ Old time-stepping methods in new ways
✓ Understanding and exploiting the interplay between algorithms and new architectures

can lead to substantial computational advances.

Changing the methods from toys to tools

Thank you !

Jan.Hesthaven@Brown.edu
http://www.cfm.brown.edu/people/jansh/