DG-FEM for PDE's
Lecture 7

Jan S Hesthaven
Brown University
Jan.Hesthaven@Brown.edu
A brief overview of what’s to come

- Lecture 1: Introduction and DG in 1D
- Lecture 2: Implementation and numerical aspects
- Lecture 3: Insight through theory
- Lecture 4: Nonlinear problems
- Lecture 5/6: 2D DG-FEM, implementation and applications
- Lecture 7: Higher order/Global problems
- Lecture 8: 3D and advanced topics
Lecture 7

✓ Let’s briefly recall what we know
✓ Brief overview of multi-D analysis
✓ Part I: Time-dependent problems
  ✓ Heat equations
  ✓ Extensions to higher order problems
✓ Part II: Elliptic problems
  ✓ Different formulations
  ✓ Stabilization
✓ Solvers and application examples
Let's summarize

We have a thorough understanding of 1st order problems

✓ For the linear problem, the error analysis and convergence theory is essentially complete.
✓ The theoretical support for DG for conservation laws is very solid.
✓ Limiting is perhaps the most pressing open problem
✓ The extension to 2D is fairly straightforward
✓ .... and we have a nice and flexible way to implement it all

Time to move beyond the 1st order problem
The heat equation

Let us consider the heat equation

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad x \in [0, 2\pi], \quad u(x, t) = e^{-t} \sin(x).
\]

We can be tempted to write this as

\[
\frac{\partial u}{\partial t} - \frac{\partial}{\partial x} u_x = 0,
\]

and then just use our standard approach

\[
v^k_h = D_r u^k_h, \quad M^k \frac{d u^k_h}{d t} - S v^k_h = - \int_{\partial D^k} \hat{n} \cdot (v^k_h - v^*) \ell^k(x) \, d x,
\]

Given the nature of the problem, a central flux seems reasonable

\[
v^* = \{\{v_h\}\}
\]
The heat equation

Let's see what happens when we run it

<table>
<thead>
<tr>
<th>$N \backslash K$</th>
<th>10</th>
<th>20</th>
<th>40</th>
<th>80</th>
<th>160</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.27E-1</td>
<td>4.34E-1</td>
<td>4.37E-1</td>
<td>4.38E-1</td>
<td>4.39E-1</td>
</tr>
<tr>
<td>2</td>
<td>5.00E-1</td>
<td>4.58E-1</td>
<td>4.46E-1</td>
<td>4.43E-1</td>
<td>4.42E-1</td>
</tr>
<tr>
<td>4</td>
<td>1.68E-1</td>
<td>1.37E-1</td>
<td>1.28E-1</td>
<td>1.26E-1</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>7.46E-3</td>
<td>8.60E-3</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

It does not work!

It is weakly unstable
The heat equation

We need a new idea -- consider

\[ \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} a(x) \frac{\partial u}{\partial x}, \]

We know that DG is good for 1st order systems.

Since \( a(x) > 0 \) we can write this as

\[ \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \sqrt{aq}, \quad q = \sqrt{a} \frac{\partial u}{\partial x}, \]

Now follow our standard approach

\[
\begin{bmatrix}
  u(x, t) \\
  q(x, t)
\end{bmatrix}
\approx
\begin{bmatrix}
  u_h(x, t) \\
  q_h(x, t)
\end{bmatrix}
= \bigoplus_{k=1}^K \begin{bmatrix}
  u_h^k(x, t) \\
  q_h^k(x, t)
\end{bmatrix}
= \bigoplus_{k=1}^K \sum_{i=1}^{N_p} \begin{bmatrix}
  u_h^k(x_i, t) \\
  q_h^k(x_i, t)
\end{bmatrix} \ell_i^k(x),
\]

Friday, July 23, 2010
The heat equation

Treating this as a 1st order system we have

\[ M^k \frac{d u_h^k}{dt} = \tilde{S} \sqrt{a} q_h^k - \int_{\partial D^k} \hat{n} \cdot ((\sqrt{a q_h^k}) - (\sqrt{a q_h^k}^*) \ell^k(x) \, dx, \]

\[ M^k q_h^k = S \sqrt{a} u_h^k - \int_{\partial D^k} \hat{n} \cdot (\sqrt{a u_h^k} - (\sqrt{a u_h^k}^*) \ell^k(x) \, dx, \]

or the corresponding weak form

\[ M^k \frac{d u_h^k}{dt} = -(S \sqrt{a})^T q_h^k + \int_{\partial D^k} \hat{n} \cdot (\sqrt{a q_h^k}^*) \ell^k(x) \, dx, \]

\[ M^k q_h^k = -(\tilde{S} \sqrt{a})^T u_h^k + \int_{\partial D^k} \hat{n} \cdot (\sqrt{a u_h^k}) \ell(x) \, dx. \]

Here

\[ \tilde{S} \sqrt{a} = \int_{D^k} \ell_i^k(x) \frac{d \sqrt{a(x) \ell_j^k(x)}}{dx} \, dx, \quad S \sqrt{a} = \int_{D^k} \sqrt{a(x) \ell_i^k(x)} \frac{d \ell_j^k(x)}{dx} \, dx. \]
The heat equation

How do we choose the fluxes?

\[
\begin{align*}
(\sqrt{aq_h})^* &= f((\sqrt{aq_h})^-, (\sqrt{aq_h})^+, (\sqrt{au_h})^-, (\sqrt{au_h})^+), \\
(\sqrt{au_h})^* &= g((\sqrt{aq_h})^-, (\sqrt{aq_h})^+, (\sqrt{au_h})^-, (\sqrt{au_h})^+). 
\end{align*}
\]

\[
\mathcal{M}^k \frac{d\mathbf{u}^k_h}{dt} = \tilde{S}^{\sqrt{a}} \mathbf{q}^k_h - \int_{\partial D^k} \hat{n} \cdot [(\sqrt{aq_h}^k) - (\sqrt{aq_h}^k)^*] \ell^k(x) \, dx,
\]

\[
\mathcal{M}^k \mathbf{q}_h^k = S^{\sqrt{a}} \mathbf{u}_h^k - \int_{\partial D^k} \hat{n} \cdot [(\sqrt{au_h}^k) - (\sqrt{au_h}^k)^*] \ell^k(x) \, dx,
\]

Problem: Everything couples -- loss of locality

However, if we restrict it as

\[
\begin{align*}
(\sqrt{aq_h})^* &= f((\sqrt{aq_h})^-, (\sqrt{aq_h})^+, (\sqrt{au_h})^-, (\sqrt{au_h})^+), \\
(\sqrt{au_h})^* &= g((\sqrt{au_h})^-, (\sqrt{au_h})^+),
\end{align*}
\]

we can eliminate q-variable locally
The heat equation

Given the nature of the heat-equation, a natural flux could be central fluxes

\[(\sqrt{aq_h})^* = \{\sqrt{aq_h}\}, \quad (\sqrt{au_h})^* = \{\sqrt{au_h}\}.\]

But is it stable?

Computing the local energy in a single element yields

\[
\frac{1}{2} \frac{d}{dt} \|u_h\|^2_D + \|q_h\|^2_D + \Theta_r - \Theta_l = 0,
\]

\[
\Theta = \sqrt{au_h}q_h - (\sqrt{aq_h})^*u_h - (\sqrt{au_h})^*q_h.
\]

\[(\sqrt{aq_h})^* = \sqrt{a}\{q_h\}, \quad (\sqrt{au_h})^* = \sqrt{a}\{u_h\}.\]

\[\Theta_r = -\frac{\sqrt{a}}{2} (u_h^+ q_h^- + u_h^- q_h^+).\]

\[\frac{1}{2} \frac{d}{dt} \|u_h\|^2_{\Omega,h} + \|q_h\|^2_{\Omega,h} = 0, \quad \text{Stability}\]
The heat equation

So this is stable!

How about boundary conditions

**Dirichlet**  
\[ u_h^+ = -u_h^- , \quad q_h^+ = q_h^- \]  
\[ \begin{align*}  
\{\{u_h\}\} &= 0, \quad [u_h] = 2\hat{n}^- u_h^-  
\{\{q_h\}\} &= q_h^- , \quad [q_h] = 0.
\end{align*} \]

**Neumann**  
\[ u_h^+ = u_h^- , \quad q_h^+ = -q_h^- \]  
\[ \begin{align*}  
\{\{u_h\}\} &= u_h^- , \quad [u_h] = 0  
\{\{q_h\}\} &= 0, \quad [q_h] = 2\hat{n}^- q_h^- .
\end{align*} \]

**Inhomogeneous BC**  
\[ u_h^+ = -u_h^- + 2f(t) , \quad q_h^+ = q_h^- , \]

... and likewise for Neumann
The heat equation

Back to the example

Looks good -

.. but an even/odd pattern

**Theorem 7.3.** Let \( \varepsilon_u = u_h - u \) and \( \varepsilon_q = q_h - q \) signify the pointwise errors for the heat equation with periodic boundaries and a constant coefficient \( a(x) \), computed with Eq. (7.1) and central fluxes. Then

\[
\| \varepsilon_u(T) \|_{\Omega,h}^2 + \int_0^T \| \varepsilon_q(s) \|_{\Omega,h}^2 \, ds \leq Ch^{2N},
\]

where \( C \) depends on the regularity of \( u, T, \) and \( N \). For \( N \) even, \( C \) is \( O(h^2) \).
The heat equation

Can we do anything to improve on this?

Recall the stability condition

\[
\frac{1}{2} \frac{d}{dt} \| u_h \|_D^2 + \| q_h \|_D^2 + \Theta_r - \Theta_l = 0,
\]

\[
\Theta = \sqrt{a} u_h q_h - (\sqrt{a} q_h)^* u_h - (\sqrt{a} u_h)^* q_h.
\]

Stable choices

\[
(\sqrt{a} u_h)^* = \{\{\sqrt{a}\} u_h^+), \quad (\sqrt{a} q_h)^* = \sqrt{a}^{-1} q_h^-.
\]

\[
(\sqrt{a} u_h)^* = \sqrt{a}^{-1} u_h^-, \quad (\sqrt{a} q_h)^* = \{\{\sqrt{a}\} q_h^+,
\]

\[
\{\{\sqrt{a} u_h}\} + \hat{\beta} \cdot [\sqrt{a} u_h], \quad (\sqrt{a} q_h)^* = \{\{\sqrt{a} q_h\} - \hat{\beta} \cdot [\sqrt{a} q_h],
\]

**Upwind/downwind - LDG flux**

\[
\hat{\beta} = \hat{n}
\]
The heat equation

Back to the example

Looks good -

.. full order restored

Theorem 7.4. Let \( \varepsilon_u = u - u_h \) and \( \varepsilon_q = q - q_h \) signify the pointwise errors for the heat equation with periodic boundaries and a constant coefficient \( a(x) \), computed with Eq. (7.1) and LDG fluxes. Then

\[
\| \varepsilon_u(T) \|^2_{\Omega,h} + \int_0^T \| \varepsilon_q(s) \|^2_{\Omega,h} \, ds \leq C h^{2N+2},
\]

where \( C \) depends on the regularity of \( u, T, \) and \( N \).
Higher order and mixed problems

We can now mix and match what we know

Consider

\[
\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = \frac{\partial}{\partial x} a(x) \frac{\partial u}{\partial x},
\]

and rewrite as

\[
\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} (f(u) - \sqrt{a}q) = 0, \quad (f(u) - \sqrt{a}q)^* \quad \text{with} \quad q = \sqrt{a} \frac{\partial u}{\partial x}, \quad (\sqrt{au_h})^*
\]

Now choose fluxes as we know how

\[
f(u)^* = \{\{f(u)\}\} + \frac{C}{2} [u], \quad C \geq \max |f'(u)|.
\]

\[
(\sqrt{au_h})^* = \{\{\sqrt{a}\}\} u_h^+, \quad (\sqrt{aq_h})^* = \sqrt{a^-} q_h^-.
\]
Higher order and mixed problems

Consider viscous Burgers equation

\[
\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} \left( \frac{u^2}{2} \right) = \varepsilon \frac{\partial^2 u}{\partial x^2}, \quad x \in [-1, 1],
\]

\[
u(x, t) = -\tanh \left( \frac{x + 0.5 - t}{2\varepsilon} \right) + 1.
\]
Higher order and mixed problems

Consider the 3rd order dispersive equation

\[
\frac{\partial u}{\partial t} = \frac{\partial^3 u}{\partial x^3}.
\]

Which boundary conditions do we need?

\[
\frac{1}{2} \frac{d}{dt} ||u||_\Omega^2 = \left[ u \frac{\partial^2 u}{\partial x^2} - \frac{1}{2} \left( \frac{\partial u}{\partial x} \right)^2 \right]_{x_l}^{x_r}, \quad \text{must be bounded}
\]

\[
x = x_l : \text{ On } u \text{ or } \frac{\partial^2 u}{\partial x^2} \text{ and } \frac{\partial u}{\partial x},
\]

\[
x = x_r : \text{ On } u \text{ or } \frac{\partial^2 u}{\partial x^2}.
\]
Higher order and mixed problems

Write it as a 1st order system

\[
\frac{\partial u}{\partial t} = \frac{\partial q}{\partial x}, \quad q = \frac{\partial p}{\partial x}, \quad p = \frac{\partial u}{\partial x}.
\]

To choose the fluxes, we consider the energy

\[
\frac{1}{2} \frac{d}{dt} \|u_h\|^2_{D^k} = \Theta_r - \Theta_l, \quad \Theta = \frac{p_h^2}{2} - u_h q_h + u_h (q_h)^* + q_h (u_h)^* - p_h (p_h)^*.
\]

Central fluxes yields

\[
\Theta = \frac{1}{2} (u_h^+ q_h^- + u_h^- q_h^+ - p_h^- p_h^+), \quad \frac{1}{2} \frac{d}{dt} \|u_h\|^2_{D^k} = 0
\]

Alternative LDG-flux

\[
(u_h)^* = u_h^-, \quad (q_h)^* = q_h^+, \quad (p_h)^* = p_h^-.
\]
Higher order and mixed problems

Consider

\[ \frac{\partial u}{\partial t} = \frac{\partial^3 u}{\partial x^3}, \quad x \in [-1, 1], \]

\[ u(x, t) = \cos(\pi^3 t + \pi x). \]

Convergence behavior exactly as for the 2nd order problem

Central flux

LDG flux
Higher order and mixed problems

Few comments

✓ The reformulation to a system of 1st order problems is entirely general for any order operator

✓ When combined with other operators, one chooses fluxes for each operator according to the analysis.

✓ The biggest problem is cost -- a 2nd order operator require two derivates rather than one.

✓ There are alternative ‘direct’ ways but they tend to be problem specific
What about the time step?

For 1st order problems we know

\[ \Delta t \leq C \frac{h}{aN^2} \]

This gets worse -

\[ \Delta t \leq C \left( \frac{h}{N^2} \right)^p \]

\( p = \) order of operator

Options:
- ✓ Local time stepping
- ✓ Implicit time stepping
Lecture 7

✓ Let’s briefly recall what we know

✓ Brief overview of multi-D analysis

✓ Part I: Time-dependent problems
  ✓ Heat equations
  ✓ Extensions to higher order problems

✓ Part II: Elliptic problems
  ✓ Different formulations
  ✓ Stabilization
  ✓ Solvers and application examples
Elliptic problems

Now we could consider solving a problem like

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - f(x),$$

However, if we are interested in the steady state we may be better off considering

$$\frac{\partial^2 u}{\partial x^2} = f(x),$$

We can use any of the methods we just derived to obtain the linear system

$$Au_h = f_h,$$
Elliptic problems

Assume we use a central flux.

When we try to solve we discover that $A$ is singular!

![Graphs showing eigenvalues and eigenvectors for different values of N and K.]

- $N=1, K=6$
- $N=4, K=6$
- $N=2, K=6$
Elliptic problems

What is happening?

The discontinuous basis is too rich -- it allows one extra null vector:

A local null vector with $\{\{u\}\}=0$

What can we do?

Change the flux by penalizing this mode

$$q^* = \{\{q\}\} - \tau [u], \quad u^* = \{\{u\}\}.$$  

The flexibility of DG shows its strength!
Elliptic problems

Does it work?

\[
\frac{d^2 u}{dx^2} = -\sin(x), \quad x \in [0, 2\pi], \quad u(0) = u(2\pi) = 0.
\]

What about the other flux - the LDG flux?
Elliptic problems

Consider the stabilized LDG flux

\[ q_h^* = \{\{q_h\}\} + \hat{\beta} \cdot [q_h] - \tau [u_h], \quad u_h^* = \{\{u_h\}\} - \hat{\beta} \cdot [u_h], \]

Works fine as expected - but we also note that A is much more sparse!
Elliptic problems

Why is one more sparse than the other?

Consider the N=0 case

\[ q_h^*(q_h^k, q_h^{k+1}, u_h^k, u_h^{k+1}) - q_h^*(q_h^k, q_h^{k-1}, u_h^k, u_h^{k-1}) = h f_h^k, \]
\[ u_h^*(u_h^k, u_h^{k+1}) - u_h^*(u_h^k, u_h^{k-1}) = h g_h^k. \]

Using the central flux yields

\[ q_h^*(q_h^-, q_h^+, u_h^-, u_h^+) = \{\{q_h\}\} - \tau [u_h], \quad u_h^*(u_h^-, u_h^+) = \{\{u_h\}\}, \]
\[ \frac{u_h^{k+2} - 2u_h^k + u_h^{k-2}}{(2h)^2} + \tau \frac{u_h^{k+1} - u_h^{k-1}}{h} = f_h^k. \]

Using the LDG flux yields

\[ q_h^*(q_h^-, q_h^+, u_h^-, u_h^+) = q_h^- - \tau [u_h], \quad u_h^*(u_h^-, u_h^+) = u_h^+, \]
\[ \frac{u_h^{k+1} - 2u_h^k + u_h^{k-1}}{h^2} + \tau \frac{u_h^{k+1} - u_h^{k-1}}{h} = f_h^k. \]
Elliptic problems

The sparsity is a good thing -- but it comes at a price

\[ \kappa(A_{LDG}) \approx 2\kappa(A_C); \]

We seek a flux balancing sparsity and conditioning?

\[ q_h^* = \{(u_h)_x\} - \tau[u_h], \quad u_h^* = \{u_h\}. \]

Internal penalty flux

\[ \kappa(A_C) \approx \kappa(A_{IP}); \]

Mission accomplished
Elliptic problems

Remaining question: How do you choose $\tau$?

The analysis shows that:

✓ For the central flux, $\tau > 0$ suffices
✓ For the LDG flux, $\tau > 0$ suffices
✓ For the IP flux, one must require

$$\tau \geq C \frac{(N + 1)^2}{h}, \quad C \geq 1,$$

These suffices to guarantee stability, but they may not give the best accuracy

Generally, a good choice is

$$\tau \geq C \frac{(N + 1)^2}{h}, \quad C \geq 1,$$
Elliptic problems

What can we say more generally?

Consider

$$-\nabla^2 u(x) = f(x), \quad x \in \Omega,$$

Discretized as

$$-\nabla \cdot q = f, \quad q = \nabla u.$$

Using one of the fluxes

<table>
<thead>
<tr>
<th>Flux Type</th>
<th>$u_h^*$</th>
<th>$q_h^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central flux</td>
<td>${{u_h}}$</td>
<td>${{q_h}} - \tau[{u_h}]$</td>
</tr>
<tr>
<td>Local DG flux (LDG)</td>
<td>${{u_h}} + \beta[{u_h}]$</td>
<td>${{q_h}} - \beta[{q_h}] - \tau[{u_h}]$</td>
</tr>
<tr>
<td>Internal penalty flux (IP)</td>
<td>${{u_h}}$</td>
<td>${{\nabla u_h}} - \tau[{u_h}]$</td>
</tr>
</tbody>
</table>
Elliptic problems

For the 3 discrete systems, one can prove (see text)

✓ They are all symmetric for any $N$
✓ They are all invertible provided stabilization is used
✓ The discretization is consistent
✓ The adjoint problem is consistent
✓ They have optimal convergence in $L^2$

Many of these results can be extended to more general problems (saddle-point, non-coercive etc)

There are other less used fluxes also
Solving the systems

After things are discretized, we end up with

\[ Au_h = f_h \]

We can solve this in two different ways

✓ Direct methods

✓ Iterative methods

The ‘right’ choice depends on things such as size, speed, sparsity etc
Solving the systems

Direct methods are ‘LU’ based

\[
\begin{align*}
\textgreater \textgreater [L, U] &= \text{lu}(A); \\
\textgreater \textgreater u &= U \backslash (L \backslash f);
\end{align*}
\]

Example:

\[
\nabla^2 u = f(x, y) = \left( (16 - n^2) r^2 + (n^2 - 36) r^4 \right) \sin (n\theta), \quad x^2 + y^2 \leq 1,
\]

\[
\begin{aligned}
&: n = 12, \quad r = \sqrt{x^2 + y^2}, \quad \theta = \arctan(y, x) \\
\end{aligned}
\]

K=512
N=4
7680 DoF
Solving the systems

Fig. 7.14. On the left, we show the sparsity pattern of the discrete Poisson matrix before reverse Cuthill-McKee reordering. On the right, we show the sparsity pattern after the reordering.

To reduce the amount of fill in for the LU-factors, one can explore reordering of the degrees of freedom to minimize the bandwidth of the matrix as much as possible. A powerful approach for this is the reverse Cuthill-McKee numbering system [125]. This is a built-in method in Matlab and provides a permutation vector for a given sparse symmetric matrix to significantly reduce the distance of any nonzero entry from the diagonal. To illustrate the effect of this reordering, we contrast in Fig. 7.14 the sparsity patterns of $A$ before and after reordering.

To take advantage of this in the solution of the linear system, one can use the following approach in which a permutation vector is computed, applied to the row and column swaps to $A$, permute $f$, solve the two systems, and permute the solution:

\[
\begin{align*}
&P = \text{symrcm}(A); \\
&A = A(P,P); \\
&\text{rhs} = \text{rhs}(P); \\
&[L,U] = \text{lu}(A); \\
&u = U \backslash (L \backslash f); \\
&u(P) = u;
\end{align*}
\]

This simple procedure reduces the number of nonzeros in the LU-factors to 3,802,198 (i.e., a reduction of more than 50%). All subsequent tests use the reordered $A$ matrix.

We can further reduce the storage requirements by taking advantage of $A$ being a positive definite symmetric matrix and use a Cholesky factorization; that is, find the matrix $C$ such that $A = C^T C$ using...
Solving the systems

Re-ordering:

\[
\begin{align*}
&\text{>> P = symrcm(A);} \\
&\text{>> A = A(P,P);} \\
&\text{>> rhs = rhs(P);} \\
&\text{>> [L,U] = lu(A);} \\
&\text{>> u = U \backslash (L \backslash f);} \\
&\text{>> u(P) = u;}
\end{align*}
\]

.. but A is SPD:

\[
\begin{align*}
&\text{A = } C^T C \\
&\text{Cholesky decomp}
\end{align*}
\]

\[
\begin{align*}
&\text{>> C = chol(A);} \\
&\text{>> u = C \backslash (C' \backslash f);} \\
&\text{1.9m extra non-zero entries in C}
\end{align*}
\]
Solving the systems

If the problem is too large, iterative methods are the only choice

>>> ittol = 1e-8; maxit = 1000;
>>> u = pcg(A, f, ittol, maxit);

Example requires 818 iterations - 100 times slower than LU!

Solution: Preconditioning

\[ C^{-1} A u_h = C^{-1} f_h, \]
Solving the systems

How to choose the preconditioning?

.. more an art than a science!

Example - Incomplete Cholesky Preconditioning

```
>> ittol = 1e-8; maxit = 1000;
>> Cinc = cholinc(OP, '0')
>> u = pcg(A, f, ittol, maxit, Cinc', Cinc);
```

138 iterations - but still 50 times slower

```
>> ittol = 1e-8; maxit = 1000; droptol = 1e-4;
>> Cinc = cholinc(A, droptol);
>> u = pcg(A, f, ittol, maxit, Cinc', Cinc);
```

17 iterations - only 2 times slower
Solving the systems

Choosing fast and efficient linear solvers is not easy -- but there are many options

✓ **Direct solvers**
  ✓ MUMPS (multi-frontal parallel solver)
  ✓ SuperLU (fast parallel direct solver)

✓ **Iterative solvers**
  ✓ Trilinos (large solver/precon library)
  ✓ PETSc (large solver/precon library)

Very often you have to try several options and combinations to find the most efficient and robust one(s)
A couple of examples

So far we have seen lots of theory and “homework” problems.

To see that it also works for more complex problems - but still 2D - let us look at a few examples

✓ Incompressible Navier-Stokes
✓ Boussinesq problems
Incompressible fluid flow

Time-dependent Navier-Stokes equations

\[ \frac{\partial u}{\partial t} + (u \cdot \nabla) u = -\nabla p + \nu \nabla^2 u, \quad x \in \Omega, \]
\[ \nabla \cdot u = 0, \]

Written on conservation form

\[ \frac{\partial u}{\partial t} + \nabla \cdot \mathcal{F} = -\nabla p + \nu \nabla^2 u, \quad \mathcal{F} = [F_1, F_2] = \begin{bmatrix} u^2 & uv \\ uv & v^2 \end{bmatrix}. \]
\[ \nabla \cdot u = 0, \]

Solved by stiffly stable time-splitting and pressure projection

- Water
- Low speed
- Bioflows
- etc

Friday, July 23, 2010
Incompressible fluid flow

The basics are

\[
N_x(u) = \nabla \cdot F_1 = \frac{\partial (u^2)}{\partial x} + \frac{\partial (uv)}{\partial y}, \quad N_y(u) = \nabla \cdot F_2 = \frac{\partial (uv)}{\partial x} + \frac{\partial (v^2)}{\partial y},
\]

.. and then take an inviscous time step

\[
\frac{\gamma_0 \tilde{u} - \alpha_0 u^n - \alpha_1 u^{n-1}}{\Delta t} = -\beta_0 N(u^n) - \beta_1 N(u^{n-1}).
\]

The pressure is computed to ensure incompressibility

\[
\frac{\tilde{u} - \check{u}}{\Delta t} = -\nabla p^{n+1}. \quad -\nabla^2 p^{n+1} = -\frac{\gamma_0}{\Delta t} \nabla \cdot \check{u}.
\]

.. and the viscous part is updated

\[
\gamma_0 \left( \frac{u^{n+1} - \tilde{u}}{\Delta t} \right) = \nu \nabla^2 u^{n+1}.
\]
Incompressible fluid flow

Kovasznay solution

<table>
<thead>
<tr>
<th>K</th>
<th>N</th>
<th>$t_{C_{d}}$</th>
<th>$C_{d}$</th>
<th>$t_{C_{l}}$</th>
<th>$C_{l}$</th>
<th>$\Delta p (t = 8)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>115</td>
<td>6</td>
<td>3.843263</td>
<td>2.952914</td>
<td>5.6742937</td>
<td>0.4966074</td>
<td>-0.1095664</td>
</tr>
<tr>
<td>460</td>
<td>6</td>
<td>3.936751</td>
<td>2.950930</td>
<td>5.6930431</td>
<td>0.4778757</td>
<td>-0.116310</td>
</tr>
<tr>
<td>236</td>
<td>8</td>
<td>3.9438595</td>
<td>2.941719</td>
<td>5.6990205</td>
<td>0.4879853</td>
<td>-0.119122</td>
</tr>
<tr>
<td>236</td>
<td>10</td>
<td>3.9370396</td>
<td>2.9545659</td>
<td>5.6927772</td>
<td>0.4789706</td>
<td>-0.116177</td>
</tr>
<tr>
<td>189</td>
<td>N/A</td>
<td>3.93625</td>
<td>2.950921575</td>
<td>5.693125</td>
<td>0.47795</td>
<td>-0.1116</td>
</tr>
</tbody>
</table>

von Karman flow

<table>
<thead>
<tr>
<th>N</th>
<th>$h$</th>
<th>$h/2$</th>
<th>$h/4$</th>
<th>Rate</th>
<th>$h$</th>
<th>$h/2$</th>
<th>$h/4$</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.32E+00</td>
<td>7.05E-01</td>
<td>1.23E-01</td>
<td>1.71</td>
<td>3.13E+00</td>
<td>1.53E+00</td>
<td>3.48E-01</td>
<td>1.59</td>
</tr>
<tr>
<td>2</td>
<td>5.01E-01</td>
<td>9.67E-02</td>
<td>1.45E-02</td>
<td>2.55</td>
<td>1.47E+00</td>
<td>2.54E-01</td>
<td>2.08E-02</td>
<td>3.07</td>
</tr>
<tr>
<td>3</td>
<td>2.41E-01</td>
<td>2.74E-02</td>
<td>1.89E-03</td>
<td>3.49</td>
<td>5.02E-01</td>
<td>2.79E-02</td>
<td>2.42E-03</td>
<td>3.85</td>
</tr>
<tr>
<td>4</td>
<td>6.40E-02</td>
<td>3.34E-03</td>
<td>1.00E-04</td>
<td>4.66</td>
<td>9.31E-02</td>
<td>8.87E-03</td>
<td>2.02E-04</td>
<td>4.42</td>
</tr>
<tr>
<td>5</td>
<td>1.87E-02</td>
<td>6.92E-04</td>
<td>8.96E-06</td>
<td>5.51</td>
<td>6.15E-02</td>
<td>1.02E-03</td>
<td>1.89E-05</td>
<td>5.84</td>
</tr>
<tr>
<td>6</td>
<td>9.07E-03</td>
<td>5.41E-05</td>
<td>6.93E-07</td>
<td>6.84</td>
<td>1.23E-02</td>
<td>2.17E-04</td>
<td>2.63E-06</td>
<td>6.09</td>
</tr>
<tr>
<td>7</td>
<td>1.43E-03</td>
<td>1.37E-05</td>
<td>4.03E-08</td>
<td>7.56</td>
<td>4.73E-03</td>
<td>4.67E-05</td>
<td>1.29E-07</td>
<td>7.58</td>
</tr>
<tr>
<td>8</td>
<td>5.91E-04</td>
<td>1.02E-06</td>
<td>7.17E-09</td>
<td>8.16</td>
<td>1.52E-03</td>
<td>3.54E-06</td>
<td>1.97E-08</td>
<td>8.12</td>
</tr>
</tbody>
</table>
Fluid-structure interaction

Boussinesq modeling
The basis assumption of this approach is to approximate the vertical variation using an expansion in $z$. 

\[-d(x)\]
Fluid-structure interaction

Under certain assumptions, the proper model (a high-order Boussinesq model) becomes

\[
\begin{align*}
\partial_t \tilde{U} + \nabla \left( g\eta + \frac{1}{2} \left( \tilde{U} \cdot \tilde{U} - \tilde{w}^2 (1 + \nabla \eta \cdot \nabla \eta) \right) \right) &= 0, \\
\partial_t \eta - \tilde{w} + \nabla \eta \cdot \left( \tilde{U} - \tilde{w} \nabla \eta \right) &= 0,
\end{align*}
\]

\[
\begin{bmatrix}
\tilde{U} \\
\tilde{V} \\
0
\end{bmatrix} = \begin{bmatrix}
A_1 - \partial_x \eta \cdot B_{11} & -\partial_x \eta \cdot B_{12} & B_{11} + \partial_x \eta \cdot A_1 \\
-\partial_y \eta \cdot B_{11} & A_1 - \partial_y \eta \cdot B_{12} & B_{12} + \partial_y \eta \cdot A_1 \\
A_{01} + S_1 & A_{02} + S_2 & B_0 + S_{03}
\end{bmatrix} \begin{bmatrix}
\hat{u}^* \\
\hat{v}^* \\
\hat{w}^*
\end{bmatrix},
\]

\[
\tilde{w} = -B_{11} \hat{u}^* - B_{12} \hat{v}^* + A_1 \hat{w}^*.
\]
Fluid-structure interaction

Where we have high-order derivates since

\[ A_{01} = \lambda \partial_x + \gamma_3 \lambda^3 (\partial_{xxx} + \partial_{xyy}) + \gamma_5 \lambda^5 (\partial_{xxxxx} + 2\partial_{xxxyy} + \partial_{xyyyy}), \]
\[ A_{02} = \lambda \partial_y + \gamma_3 \lambda^3 (\partial_{xyy} + \partial_{yyy}) + \gamma_5 \lambda^5 (\partial_{xxxxy} + 2\partial_{xxyyy} + \partial_{yyyyy}), \]
\[ A_1 = 1 - \alpha_2 (\partial_{xx} + \partial_{yy}) + \alpha_4 (\partial_{xxxx} + 2\partial_{xxxy} + \partial_{xyyy}), \]
\[ B_0 = 1 + \gamma_2 \lambda^2 (\partial_{xx} + \partial_{yy}) + \gamma_4 \lambda^4 (\partial_{xxxx} + 2\partial_{xxxy} + \partial_{xyyy}), \]
\[ B_{11} = \beta_1 \partial_x - \beta_3 (\partial_{xxx} + \partial_{xyy}) + \beta_5 (\partial_{xxxxx} + 2\partial_{xxxyy} + \partial_{xyyyy}), \]
\[ B_{12} = \beta_1 \partial_y - \beta_3 (\partial_{xyy} + \partial_{yyy}) + \beta_5 (\partial_{xxxxy} + 2\partial_{xxyyy} + \partial_{yyyyy}), \]

\[ S_1 = \partial_x d \cdot C_1, \quad C_1 = 1 - c_2 \lambda^2 (\partial_{xx} + \partial_{yy}) + c_4 \lambda^4 (\partial_{xxxx} + 2\partial_{xxxy} + \partial_{xyyy}). \]
\[ S_2 = \partial_y d \cdot C_1, \]

A bit on the complicated side!
A couple of 2D(1D) tests

Submerged bar (K=110, P=8) - comparison with experimental data

Friday, July 23, 2010
A couple of 3D(2D) tests

McCamy & Fuchs (1954)

DG-FEM solution:
$ka=\pi, \ k_d=1.0,\ P=4, \ K=1261,\ t=0.03s$
Remarks

We are done with all the basic now! -- and we have started to see it work for us

What we need to worry about is:

✓ The need for 3D
✓ The need for speed
✓ Software beyond Matlab

Last frontier!