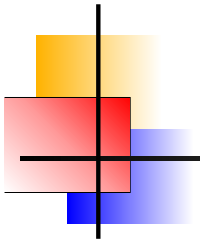
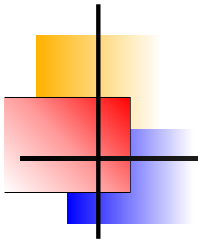


A Technique for the Quantitative Assessment of the Solution Quality on Particular Finite Elements in COMSOL Multiphysics

Matthias K. Gobbert

Department of Mathematics and Statistics
University of Maryland, Baltimore County
gobbert@math.umbc.edu
http://www.math.umbc.edu/~gobbert

October 05, 2007



Problem Statement

- Problem: Assess the quality of a FEM solution quantitatively.
- Approach: Use guidance from the a priori error estimate

$$\|u - u_h\|_{L^2(\Omega)} \leq C\, h^q, \qquad \text{as } h \to 0$$

with a constant C independent of h and the convergence order $q > 0$. Here, h is the maximum side length of the elements in the triangulation.

- Concrete goal: Confirm that a given sequence of solutions on progressively refined meshes behaves as predicted by the error estimate.

Example of a Convergence Study

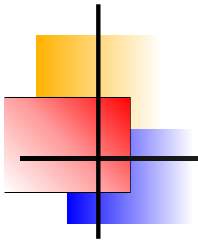
If we have $E_r := \|u - u_h\|_{L^2(\Omega)} \approx C\, h^q$ on refinement level r,

- then $E_r := \|u - u_h\|_{L^2(\Omega)} \approx C\, h^q$ and $E_{r-1} = \|u - u_{2h}\|_{L^2(\Omega)} \approx C\, 2^q\, h^q$,
- then $R_r := E_{r-1}/E_r \approx 2^q$ and $Q_r := \log_2(R_r) = q$, that is, Q_r is a computable estimate for q.

Example table of results:

r	DOF	E_r	R_r	Q_r
0	25	3.74e-003		
1	85	1.01e-003	3.71	1.89
2	313	2.59e-004	3.90	1.96
3	1201	6.51e-005	3.98	1.99
4	4705	1.60e-005	4.07	2.02
5	18625	3.64e-006	4.40	2.14

These results indicate that $q = 2$ for the PDE and FEM used.



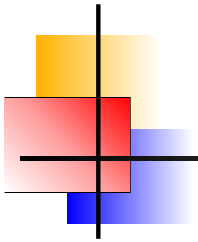
Application to Linear Lagrange Elements

For instance, for linear Lagrange FEM, the a priori error estimate holds with $q = 2$.
The expected convergence rate for linear Lagrange elements is valid, provided that

- the true solution u is smooth enough,
- the domain Ω is bounded, convex, and simply connected,
- and the domain boundary $\partial\Omega$ piecewise smooth, i.e., the domain Ω can be discretized sufficiently well.

On the one hand, if a sequence of solutions does not satisfy $q = 2$, then the PDE data and its domain may not be smooth enough.

On the other hand, the test allows to assess how badly some of the theoretical assumptions are violated and whether their violation is a serious problem.



Outline

This talk will show how to do obtain a table as above using COMSOL Multiphysics and COMSOL Script for linear Lagrange elements, without a known true solution.

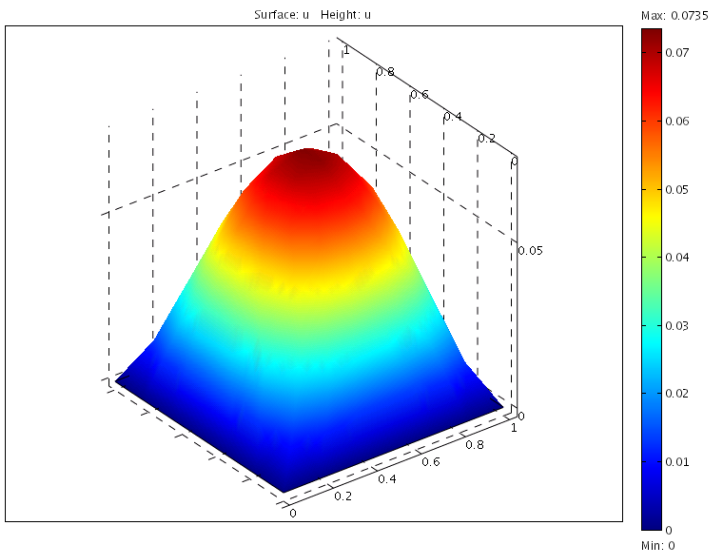
- Solve the desired PDE in the GUI of COMSOL Multiphysics.
- Set up a function m-file `getfem.m` that solves the desired PDE for a chosen refinement level as input argument.
- Write a driver script in COMSOL Script that calls `getfem` for $r = 0, 1, \ldots, r_{\text{max}}$
- Compute the error $\|u - u_h\|_{L^2(\Omega)}$ using as reference solution for u the FEM solution on the finest mesh.
- Extensions, Limitations, and Alternatives

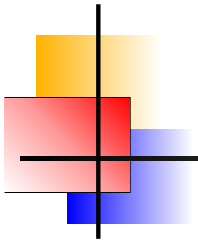
Example Problem

Use the default problem in the PDE Modes (stationary coefficient form) of COMSOL Multiphysics:

$$
\begin{aligned}
-\triangle u &= f \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \partial\Omega,
\end{aligned}
$$

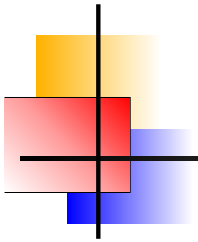
with $f \equiv 1$ on the unit square $\Omega = (0,1)^2 \subset \mathbb{R}^2$.





Create m-file `getfem.m`

- Solve desired problem in the GUI of COMSOL Multiphysics, using linear Lagrange elements. For best benefit:
  - start COMSOL Multiphysics from scratch (to decrease the clutter in the m-file),
  - use coarsest mesh possible in Mesh → Free Mesh Parameters (so we can refine as often as possible),
  - refine the mesh once (to get the `meshrefine` command into the m-file).
- Save As an m-file, which we will call from COMSOL Script.



Edit the m-file `getfem.m`

Edit m-file as follows:

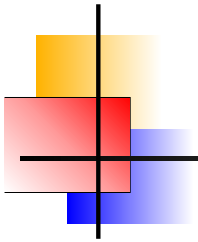
- Insert the function header

```
function fem = getfem (nref)
```

- Enclose the `meshrefine` command in a for-loop:

```
% Refine mesh
for nr = 1 : nref
  fem.mesh=meshrefine(fem, ...
      'mcase',0, ...
      'rmethod','regular');
end;
```

- Delete the unneeded commands at the end of the m-file (i.e., the plot commands).



The Driver Script

```
% set the max. number of refinements:
nrefmax = 6;

% reference solution on finest mesh:
fem_ref = getfem (nrefmax);

% obtain mass matrix for finest mesh:
fema = fem_ref;
fema.equ.c = 0;
fema.equ.a = 1;
fema.xmesh = meshextend (fema);
[Mass,L,M,N] = assemble (fema);
p_ref = fema.mesh.p;
clear fema; % clear to save memory

% interp. ref. sol. to ref. mesh:
u_ref=postinterp(fem_ref,'u',p_ref);
```

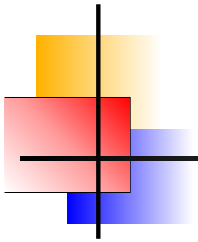
```
for nref = 0 : nrefmax-1
  fem = getfem (nref);
  D(nref+1) = length(fem.sol.u);

  % interpolate sol. to ref. mesh:
  u_int=postinterp(fem,'u',p_ref);

  % compute error as a column vector:
  e = u_ref(:) - u_int(:);

  % compute L2-norm of nodal error:
  E(nref+1) = sqrt(e'*Mass*e);
end;

R = E(1:nrefmax-1) ./ E(2:nrefmax);
Q = log2(R);
```



Heart of the Driver Script

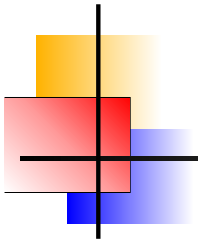
For-loop over the refinement level in right column of listing:

- compute fem for the refinement level `nref`
- save the number of DOF in vector `D`
- interpolate the FEM solution to the reference mesh
- compute the error between the interpolated solution and the reference solution
- compute and save the norm of its error

After the for-loop, compute the ratios `R` and the estimate `Q` to the convergence order. The table shows then `D`, `E`, `R`, and `Q`.

Need to explain:

- What is the reference solution and reference mesh?
- How is the norm of the error computed?



Reference Solution and Reference Mesh

Use the FEM solution on the finest mesh as the reference solution that plays the role of the true solution in the error computation:

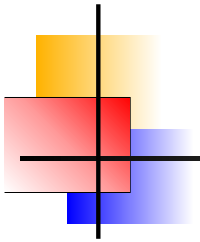
```
fem_ref = getfem (nrefmax);
```

Set up the extended mesh for the PDE $-\nabla \cdot (c\nabla u) + au = f$ with $c = 0$ and $a = 1$ on the reference mesh to compute the mass matrix $M = \texttt{Mass}$ by:

```
fema = fem_ref;
fema.equ.c = 0;
fema.equ.a = 1;
fema.xmesh = meshextend (fema);
[Mass,L,M,N] = assemble (fema);
```

Obtain the mesh points of the reference mesh:

```
p_ref = fema.mesh.p;
```

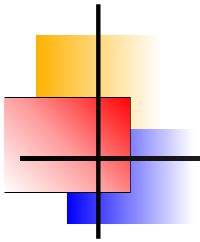


FEM Representation of the Solutions

The FEM solution on a mesh is given by the expansion

$$u_h(\mathbf{x}) = \sum_{k=1}^{N} u_k \, \varphi_k(\mathbf{x})$$

involving the FEM basis functions $\varphi_k(\mathbf{x})$, $k = 1, \ldots, N$,
where N is the number of DOFs of the mesh.
For linear Lagrange elements, the basis functions satisfy $\varphi_k(\mathbf{x}_\ell) = 1$ for $k = \ell$ and $= 0$ otherwise. That is, $u_h(\mathbf{x}_k) = u_k$ and the expansion coefficients u_k are the values of the FEM solution at the mesh points $\mathbf{x}_k$ of the triangulation.
Hence, the vectors `u_ref` and `u_int` obtained by interpolating to mesh points `p_ref` of the reference mesh hold all expansion coefficients of u and (interpolated) u_h, respectively.



Norm Computation of the Error

The error $e_h = u - u_h$ also has an expansion

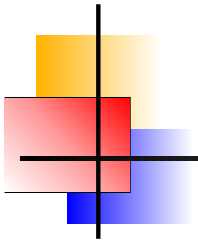
$$e_h(\mathbf{x}) = \sum_{k=1}^{N} e_k \, \varphi_k(\mathbf{x}) = \sum_{\ell=1}^{N} e_\ell \, \varphi_\ell(\mathbf{x})$$

in terms of the FEM basis functions $\varphi_k(\mathbf{x})$, and the coefficients e_k are the components of the vector `e = u_ref(:) - u_int(:)`. Therefore, apply the definition of the L^2-norm to find:

$$\|e_h\|^2_{L^2(\Omega)} = \int_\Omega e_h \, e_h \, d\mathbf{x} = \int_\Omega \left(\sum_{k=1}^{N} e_k \varphi_k(\mathbf{x}) \right) \left(\sum_{\ell=1}^{N} e_\ell \varphi_\ell(\mathbf{x}) \right) d\mathbf{x}$$

$$= \sum_{k=1}^{N} \sum_{\ell=1}^{N} e_k \left(\int_\Omega \varphi_k(\mathbf{x}) \varphi_\ell(\mathbf{x}) \, d\mathbf{x} \right) e_\ell = \sum_{k=1}^{N} \sum_{\ell=1}^{N} e_k M_{k\ell} e_\ell = e^T M e$$

with the mass matrix $M_{k\ell} = \int_\Omega \varphi_k(\mathbf{x}) \, \varphi_\ell(\mathbf{x}) \, d\mathbf{x}$, thus the norm is computed as `sqrt(e'*Mass*e)`.



The Driver Script Repeated

```
% set the max. number of refinements:
nrefmax = 6;

% reference solution on finest mesh:
fem_ref = getfem (nrefmax);

% obtain mass matrix for finest mesh:
fema = fem_ref;
fema.equ.c = 0;
fema.equ.a = 1;
fema.xmesh = meshextend (fema);
[Mass,L,M,N] = assemble (fema);
p_ref = fema.mesh.p;
clear fema; % clear to save memory

% interp. ref. sol. to ref. mesh:
u_ref=postinterp(fem_ref,'u',p_ref);
```

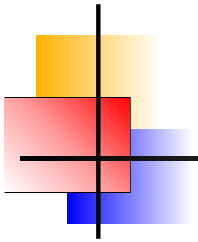
```
for nref = 0 : nrefmax-1
  fem = getfem (nref);
  D(nref+1) = length(fem.sol.u);

  % interpolate sol. to ref. mesh:
  u_int=postinterp(fem,'u',p_ref);

  % compute error as a column vector:
  e = u_ref(:) - u_int(:);

  % compute L2-norm of nodal error:
  E(nref+1) = sqrt(e'*Mass*e);
end;

R = E(1:nrefmax-1) ./ E(2:nrefmax);
Q = log2(R);
```



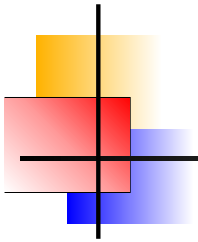
Extensions

Extensions using the same driver script due to its abstract formulation:

- other domains, PDEs, etc. instead of this example problem,
- other spatial dimensions (e.g., 3-D),
- quadrilateral elements (with linear FEM basis functions),
- transient PDEs (at selected times each).

Extensions requiring modification of the driver script:

- other application modes (name of the solution is not `u`),
- quadratic Lagrange elements (refine reference mesh once).
- systems of PDEs (apply to each component)



Alternatives and Limitations

Alternatives to and limitations of the suggested procedure:

- a posteriori error estimates (designed to measure error from computable quantities), though it is valuable to be able to check the a priori error when it is available,
- the computation of the mass matrix in other modes not the same,
- the approach cannot handle higher-degree or non-Lagrange elements $\Longrightarrow$ I am interested in suggestions how to make this general for more or all FEM in COMSOL!
- also any other suggestions for how to use COMSOL's functionality better is welcome.

The extension to higher-order elements is relevant, because the results of a convergence study indicate whether an excessive polynomial degree of the basis functions is used.