Excerpt from the Proceedings of the COMSOL Conference 2007, Boston

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, 1000 Hilltop Circle, Baltimore, MD 21250, gobbert@math.umbc.edu

Abstract: To validate the reasonableness of a numerical solution to a partial differential equation on a given mesh, a common approach is to refine the mesh, compute a solution on the finer mesh, and compare the solutions on the two meshes. Comparing graphical representations of the two solutions gives a qualitative assessment of the solution quality. In many cases though, a priori error estimates from the theory of the finite element method are available that provide quantitative predictions of the expected solution quality in terms of the mesh spacing. This note shows how to use tools available in COMSOL Multiphysics to compute numerical estimates that can confirm if the finite element method performs as predicted by the theory. The technique presented does not assume that the true solution of the PDE is known. It is applied to linear Lagrange elements here, and extensions and limitations of the technique are discussed.

Key words: Poisson equation, a priori error estimate, convergence study, mesh refinement.

1 Introduction

After successfully computing a numerical solution to a partial differential equation (PDE) with some software, say, a FEM solution with COMSOL Multiphysics, the attention typically shifts to the goal of gaining confidence in the correctness and accuracy of the computed solution. If the solution is computed on a given mesh, one common approach is to refine the mesh uniformly, re-compute the solution on the fine mesh, and compare the results on both meshes graphically. This comparison of the two solutions gives a qualitative assessment of correctness of the solution. To gain a quantitative assessment of the accuracy of the solution, one option is to use so-called a priori error estimates from the theory of the finite element method (FEM) that predict how the error should improve as the mesh is refined. By applying these results repeatedly for progressively smaller mesh spacings, one can assess if the sequence of solutions is converging as expected based on the theory.

For concreteness, consider a PDE on a spatial domain $\Omega \subset \mathbb{R}^2$ in two dimensions and let the mesh spacing h denote the maximum side length of the triangles in the mesh used to discretize the domain Ω. A standard a priori error estimate for the FEM solution u_h predicts that the norm of its error against the true solution u of the PDE satisfies

$$\|u - u_h\|_{L^2(\Omega)} \leq C\, h^q, \quad \text{as } h \to 0 \tag{1.1}$$

with a constant C independent of h and the convergence order $q > 0$. If the PDE and its domain satisfy appropriate assumptions, discussed below, the value of the exponent q in (1.1) depends on the polynomial degree of the shape functions used in the finite element space. For instance, for Lagrange elements with linear shape functions, the theory predicts $q = 2$, that is, quadratic convergence of the norm of the solution error.

The purpose of this paper is to explain how to obtain a numerical confirmation that a FEM solution computed by COMSOL Multiphysics actually behaves as predicted by (1.1) with $q = 2$ for linear Lagrange elements. Since this behavior is only present if both PDE and the FEM satisfies certain assumptions, this method can also be used to either confirm that the problem and method behave correctly or demonstrate that the problem is not as well-behaved as expected. To this end, we will compute an estimate for q from numerical results by considering a sequence of FEM so-

Table 1: Convergence study for the model problem (1.3)–(1.4) using linear Lagrange elements.

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

lutions u_h on meshes with progressively smaller h. More specifically, starting from some initial mesh, we will refine it uniformly repeatedly, which subdivides every triangle into four triangles. If h measures the maximum side length of all triangles, this procedure halves the value of h in each refinement. Then assuming that $\|u - u_h\|_{L^2(\Omega)} = C\,h^q$, the error for the next coarser mesh with mesh spacing $2h$ is $\|u - u_{2h}\|_{L^2(\Omega)} = C\,(2h)^q = 2^q\,C\,h^q$ and their ratio $\|u - u_{2h}\|_{L^2(\Omega)}/\|u - u_h\|_{L^2(\Omega)} = 2^q$. If r denotes the number of refinement levels from the initial mesh and $E_r := \|u - u_h\|_{L^2(\Omega)}$, then $E_{r-1} = \|u - u_{2h}\|_{L^2(\Omega)}$ and we can write $R_r = E_{r-1}/E_r$ for their ratio. The quantity $Q_r = \log_2(R_r)$ provides us then with a computable estimate for q in (1.1), as $h \to 0$, provided the PDE and the FEM satisfy appropriate assumptions. Table 1 lists the number of degrees of freedom (DOF) along with the quantities E_r, R_r, and Q_r for the example of a model problem, specified below. The error decreases with increasing level of mesh refinement, and the values for R_r and Q_r confirm that the PDE and FEM with linear Lagrange elements behave as (1.1) with $q = 2$, as predicted by the theory. To obtain results such as in Table 1 for any PDE, for which the true solution u is not known, one classical technique is to use a numerical solution computed on a finer mesh, here one with refinement level 6, as reference solution.

The error estimate (1.1) is a classical result and holds under appropriate assumptions on the PDE and its domain and on the finite elements both for elliptic and parabolic problems. For elliptic PDEs such as the Poisson equation $-\triangle u = f$, the classical assumption is that $f \in L^2(\Omega)$, see, e.g., [1, Corollary II.7.7]. For parabolic PDEs such as the heat equation $u_t - \triangle u = f$, the assumption $f \in L^2(\Omega)$ also suffices and (1.1) applies in the sense of the solution at every point in time, see, e.g., [3, Theorem 1.3]. Here, $L^2(\Omega)$ denotes the function space of square-integrable functions, that is, those functions v defined on the domain Ω for which $\int_\Omega v^2\,d\mathbf{x} < \infty$. The error estimate (1.1) uses its associated norm defined as

$$\|v\|_{L^2(\Omega)} := \left(\int_\Omega v^2\,d\mathbf{x}\right)^{1/2} \tag{1.2}$$

for all $v \in L^2(\Omega)$. The assumptions on the domain needed for the classical theory include that Ω be open, bounded, convex, and simply connected, and that its boundary $\partial\Omega$ be piecewise smooth. The assumption on the boundary is easiest satisfied if Ω is a polygonal domain such that it can be subdivided into a union of triangles to form a triangulation T_h with mesh spacing h as defined above.

Note that it still makes sense to employ the convergence test described in the following, even if some of these assumptions are violated, because it provides a quantitative gauge for how badly the violated assumption in fact degrades the convergence, if at all. For instance, the textbook proofs in [1, 3] require that the domain be convex and simply connected. In practice, these assumptions can often not be satisfied, e.g., for a domain with a hole, but it is observed that the FEM can still give qualitatively good results in many cases, and computable estimates such as the ones in Table 1 are useful to assess this observation quantitatively.

The remainder of this paper explains in detail how to obtain results such as in Table 1 for FEM solutions computed by COMSOL Multiphysics, without assuming that the true solution of the PDE is known. Specifically, Section 2 explains how to use the graphical user interface (GUI) of COMSOL Multiphysics to create a m-file `getfem.m` that solves the PDE under consideration for a desired refinement level r. As model problem for the m-file, we consider the default problem for the stationary PDE mode in COMSOL Multiphysics, that is, the Poisson equation with right-hand side $f \equiv 1$ with homogeneous Dirichlet boundary conditions

$$-\triangle u = 1 \quad \text{in } \Omega, \tag{1.3}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{1.4}$$

on the unit square $\Omega = (0,1)^2 \subset \mathbb{R}^2$. The m-file `getfem` can then be called repeatedly for each desired refinement level from a driver script run in COMSOL Script that computes the data needed for Table 1, as shown in Section 3. Finally, Section 4 discusses extensions and limitations of the approach presented in this paper.

2 The Function `getfem`

We first need to create an m-file `getfem` that solves the desired PDE on a mesh obtained from an initial mesh by a number of mesh refinements r. To solve, for example, the model problem (1.3)–(1.4) using the GUI of COMSOL Multiphysics, select in the Model Navigator the problem dimension 2D and COMSOL Multiphysics → PDE Modes → Coefficient Form → Stationary Analysis. In the draw mode of the GUI, draw the desired domain, here the unit square $\Omega = (0,1)^2$. (At this point, it is convenient to use Options → Zoom → Zoom Extents to re-center the GUI window.) For the example (1.3)–(1.4), all other PDE and boundary coefficients are at their default values. Also in the case that you are solving any other PDE than this model problem, I am assuming now that you have also used the GUI of COMSOL Multiphysics to obtain a preliminary solution.

To proceed towards confirming convergence of the linear Lagrange elements, ensure first that these elements are selected in COMSOL Multiphysics. To do this, select Physics → Subdomain Settings, highlight the subdomain and then select in the Element tab the elements labeled Lagrange—Linear.

Mesh (or re-mesh if you already have a mesh) the domain with a coarse initial mesh, so that we can use several refinement levels later, as follows. In Mesh → Free Mesh Parameters, choose Predefined mesh size as Extremely coarse. When I re-meshed now for the unit square $\Omega = (0,1)^2$, I have got a mesh with 36 elements. The fewer elements now, the more refinements we can perform in the convergence study and thus get more data in the table in the end.

I suggest to select Mesh → Refine Mesh once at this point, which gave me 144 triangular elements. This step is suggested to get the `meshrefine` command into the m-file that we will export later. Mesh → Mesh Statistics confirms that we have 85 degrees of freedom (DOF) as well as 85 mesh points, thus confirming the linear Lagrange elements, which have one DOF per node. Notice that this is the value for the DOF in Table 1 for refinement level $r = 1$.

Now, have COMSOL solve the problem by selecting the solution icon (the = sign) or in another way. This also causes COMSOL to display the solution. The solution in 3-D view using default settings is shown in Figure 1. (To assemble this manuscript, the plot was saved to file by File → Export → Image in TIFF format.)

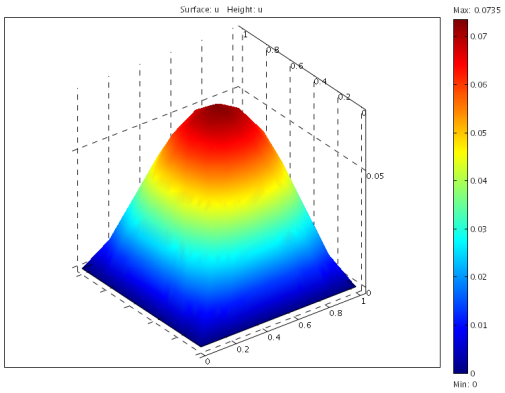

Figure 1: Solution to (1.3)–(1.4).

At this point, create the m-file by saving the entire interactive session in the GUI up to this point using File → Save As. After selecting m-file in the Files of Type field, navigate to a desired directory and provide the File Name `getfem.m`.

You can now start COMSOL Script and run the script that was just saved to confirm that you get exactly the same result as from the GUI. Specifically, either start COMSOL Script from the GUI in File → COMSOL Script, or exit the GUI and start COMSOL Script itself without the GUI. The solution log will state the number of degrees of freedom, which helps confirm, in addition to checking the plot displayed, that the problem was solved on the same mesh.

At this point, we have a script file `getfem.m` that solves the problem exactly as the GUI. We now wish to modify it to obtain a function that solves the problem for a desired refinement level $r = 0, 1, \ldots$, which is input as variable `nref`. To this end, edit the file `getfem.m` as follows:

- In the first line of the file, insert the function header

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

  This means that the function will accept `nref` as input variable and return the FEM struct `fem` to the calling driver routine.

- Search for the call to `meshrefine` in function `getfem.m`, and enclose it in a for-loop that counts `nref` many refinement levels, yielding the following code:.

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

Notice that a value of 0 for `nref` is legitimate and means that no refinement takes place, as the for-loop has an empty counter range.

- Delete (or comment out) the unneeded commands at the end of `getfem.m`, namely the line `fem0=fem` and all plot commands such as `postplot`. For the model problem (1.3)–(1.4), the call to `femstatic` that computes the FEM struct `fem` should be the final line of code in `getfem`.

After these edits, `getfem.m` is a function that we can call from a driver routine to solve the desired PDE using a mesh obtained by refining the initial mesh `nref` times. It returns the FEM struct `fem` to the calling routine, which contains all data of the problem, in particular the mesh points and the solution. The complete function `getfem.m` for the example of the model problem (1.3)–(1.4) is printed in the following. (Some line breaks and spacings of some code segments have been slightly edited here to fit the space available.)

```
function fem = getfem (nref)

% COMSOL Multiphysics Model M-file
% Generated by COMSOL 3.3
%     (COMSOL 3.3.0.405,
%     $Date: 2006/08/31 18:03:47 $)

flclear fem

% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.3';
vrsn.ext = '';
vrsn.major = 0;
vrsn.build = 405;
vrsn.rcs = '$Name:  $';
vrsn.date = ...
    '$Date: 2006/08/31 18:03:47 $';
fem.version = vrsn;

% Geometry
g1=rect2(1,1, ...
    'base','corner','pos',[0,0]);

% Analyzed geometry
clear s
s.objs={g1};
s.name={'R1'};
s.tags={'g1'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);

% Initialize mesh
fem.mesh=meshinit(fem, ...
                  'hauto',9);

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

% (Default values are not included)

% Application mode 1
clear appl
appl.mode.class = 'FlPDEC';
appl.assignsuffix = '_c';
clear prop
prop.elemdefault='Lag1';
appl.prop = prop;
clear bnd
bnd.type = 'dir';
bnd.ind = [1,1,1,1];
appl.bnd = bnd;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem);

% Solve problem
fem.sol=femstatic(fem, ...
                  'solcomp',{'u'}, ...
                  'outcomp',{'u'});
```

3 The Driver Script

The script `driver_getfem` performs the convergence study by calling the function `getfem` on progressively finer meshes, up to a maximum refinement level set in `nrefmax`, that computes the data reported in Table 1. The script is listed in its entirety here, followed by explanations of several key aspects.

```
% 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

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

% preallocate vectors:
D = repmat (NaN, [nrefmax 1]);
E = repmat (NaN, [nrefmax 1]);

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);
```

The script starts by setting `nrefmax` to the desired maximum refinement level that controls the finest mesh used. The call to `getfem` computes the FEM solution on this mesh. This reference solution is used in place of the true solution in the error $u - u_h$. This is a standard technique in convergence studies in the typical case that the true solution for the PDE is not known. For the model problem (1.3)–(1.4), the refinement level 6 leads to 74113 degrees of freedom (DOF). On a computer with an Intel Xeon 2.0 GHz CPU, the calculation on this mesh took less than 15 seconds including all startup. A memory usage of 145 MB was observed this mesh, making this the finest mesh solvable with the 512 MB of available memory.

The following commands in the driver script obtain a mass matrix and nodes of the reference mesh. The matrix needed in the norm computations later in the script has elements

$$M_{k\ell} = \int_\Omega \varphi_k(\mathbf{x})\, \varphi_\ell(\mathbf{x})\, d\mathbf{x} \tag{3.1}$$

defined in terms of the FEM basis functions $\varphi_k(\mathbf{x})$ associated with all degrees of freedom. I use COMSOL Multiphysics here to compute this matrix in variable `Mass` by assembling an auxiliary finite element discretization in the struct `fema` for a PDE with the problem coefficients specifically chosen to compute $M_{k\ell}$ in (3.1).

Next, the reference solution in the FEM struct `fem_ref` is formally interpolated to the nodes `p_ref` of the reference mesh. This amounts to a re-ordering and is necessary, because the order of components in the solution vector `fem_ref.sol.u` does not agree with the ordering of the nodes in `p_ref`.

The for-loop is the heart of the script and computes the FEM solutions for the desired refinement levels from 0 to `nrefmax-1` by calling the function `getfem` for each refinement level. For the linear Lagrange elements, the length of the solution vector is the number of DOF, which are saved in the vector `D`. After the solution in the FEM struct `fem` is interpolated to the nodes of the reference mesh, the nodal error against the reference solution is computed in vector `e`; the use of the colon `:` in the arguments of the vectors makes `e` a column vector.

To compute the L^2-norm of the error, we use the fact that a FEM solution is given in terms of the FEM basis functions $\varphi_k(\mathbf{x})$ by the expansion

$$u_h = \sum_{k=1}^{N} u_k\, \varphi_k(\mathbf{x}), \tag{3.2}$$

where N denotes the number of degrees of freedom. For linear Lagrange elements, the expansion coeffi-

cients u_k are the solution values at the nodes of the mesh. Therefore, the FEM error $e_h = u - u_h$ also has an expansion of the form (3.2) and its expansion coefficients e_k are exactly the nodal errors in the components of the vector `e`. The connection to the mass matrix with elements $M_{k\ell}$ defined above becomes clear after using the representation of e_h in the norm definition (1.2), which gives

$$\|e_h\|_{L^2(\Omega)} = \left(\sum_{k=1}^{N} \sum_{\ell=1}^{N} e_k M_{k\ell} e_\ell \right)^{1/2} . \qquad (3.3)$$

Thus the computation of the double sum can be programmed as `e'*Mass*e`, for which we need `e` to be a column vector.

Finally, the last two lines of the driver script compute the vectors `R` and `Q` from the errors in `E`. Using `format compact` and `format long`, the COMSOL Script window shows the following results for the model problem, which is the raw data for Table 1.

```
C>> driver_getfem
C>> D
D =
     25
     85
    313
   1201
   4705
  18625
C>> E
E =
     0.00373991924891
     0.00100787252430
   2.58724778510e-004
   6.50628338773e-005
   1.59880180888e-005
   3.63752316794e-006
C>> R
R =
     3.71070662088152
     3.89553922936714
     3.97653718862375
     4.06947462256785
     4.39530343879178
C>> Q
Q =
     1.89169394193255
     1.96182304322636
     1.99151266309084
     2.02484255165592
     2.13596276821933
```

4 Extensions and Limitations

We showed a procedure for estimating the convergence order of the FEM solution in the L^2-norm using COMSOL Multiphysics and COMSOL Script. We demonstrated the procedure for a stationary PDE in two spatial dimensions. The same procedure can be applied to transient PDEs at several chosen points in time, as used in [2]. It can also be readily used in other space dimensions. The a priori error estimate in (1.1) also holds for quadrilateral meshes and the proposed procedure generalizes to these elements. Due to the abstract formulation of the norm computation, these generalizations usually only involve modifications in `getfem` in the way the meshes are initialized and/or refined.

The assembly of the mass matrix as given in the driver script uses coefficient names that are specific to the PDE Modes in COMSOL Multiphysics. The approach should work in principle for other modes with appropriate changes of variable names, including the mode dependent name of the solution variable.

However, the specific procedure suggested to compute the L^2-norm of the error between the FEM approximation and a reference solution relied on the use of linear Lagrange elements to guarantee that the error has an expansion of the form (3.2) with nodal errors as expansion coefficients. The approach can be somewhat generalized to quadratic Lagrange elements by using the nodes and mass matrix associated with another uniform refinement of the auxiliary mesh. However, a better approach might exist that both uses COMSOL Multiphysics more effectively and can be used for all types of finite elements available.

References

[1] Dietrich Braess. *Finite Elements*. Cambridge University Press, third edition, 2007.

[2] Matthias K. Gobbert, Martin Kružík, and Thomas I. Seidman. Numerical approximation of a heat equation with measure-valued data. In preparation.

[3] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems*, volume 25 of the *Springer Series in Computational Mathematics*. Springer-Verlag, second edition, 2006.