Introduction to Finite Element Modelling in Geosciences: **Code Verification**

D. A. May & M. Frehner
ETH Zürich

July 10, 2014

1 Overview

The purpose of this session is to learn how to determine if your implementation of the finite element method (i.e. your Matlab code) is "working correctly". By "working correctly" we mean that your FE code is solving the partial differential equations (PDE's) accurately, and most importantly that the numerical errors associated with the discrete solution of the PDE decrease at the expected rate when the grid is refined.

2 Basics

Suppose we had a smooth, infinitely differentiable function $f(x)$, then we can locally approximate the function via an infinite Taylor series expansion

$$\begin{aligned} f(x+h) &= f(x) + h\frac{\partial f}{\partial x} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2} + \frac{h^3}{6}\frac{\partial^3 f}{\partial x^3} + \dots \\ &= \sum_{i=0}^{N} \frac{h^n}{n!}\frac{\partial^n f}{\partial x^n}, \end{aligned} \tag{1}$$

where h is a small perturbation. The more terms N we include, the more accurate the expansion is. For example, consider the truncated Taylor series expansion

$$f(x+h) = f(x) + h\frac{\partial f}{\partial x} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2} + \frac{h^3}{6}\frac{\partial^3 f}{\partial x^3}, \tag{2}$$

where all terms with powers ≥ 4 have been neglected. We will use the notation $\mathcal{O}(h^k)$ to indicate the order of accuracy of discrete solutions. We can express the approximation error in Eqn. (2) which results from dropping the terms with order $h^\alpha, \alpha \geq 4$ via

$$f(x+h) = f(x) + h\frac{\partial f}{\partial x} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2} + \frac{h^3}{6}\frac{\partial^3 f}{\partial x^3} + \mathcal{O}(h^4), \tag{3}$$

which indicates that the truncated Taylor series is fourth order accurate.

3 Errors

Consider the PDE

$$\nabla^2 u = f, \tag{4}$$

where we denote by u, the exact solution. If we apply the FE method to the above we obtain the following matrix problem

$$L^h u^h = F^h. \tag{5}$$

Here L^h is the discrete Laplacian and F^h is the discrete right hand side of the PDE. We denote by u^h the solution of the discrete problem. Note that in general, u^h will only ever by an *approximate* solution to the PDE. Accordingly, we can define the discretisation error e, as

$$e = u - u^h. \tag{6}$$

The superscript h notation is frequently used to indicate that the discrete solution is a function of the grid resolution h. In the above, e is function of space. In practice, it is more convenient to quantify the error via a single number, i.e. via some measure of the global discretisation error. Possible choices for global error measure include the L_1 measure

$$E(\Omega)_{L_1} = \int_\Omega \left| u - u^h \right| \, dV \tag{7}$$

and the L_2 measure

$$E(\Omega)_{L_2} = \sqrt{\int_\Omega \left| u - u^h \right|^2 \, dV}. \tag{8}$$

3.1 Sources of errors

We consider that there are two main sources of error in the FE implementations discussed in this course. These can be classified as *discretisation errors* or *numerical errors.* Errors associated with discretisation related to the size of the element used h, and the order of the polynomial used p. For example, the first exercise used ten ($h = 10/10$, 1D linear elements ($p = 1$). Numerical errors relate to round-off (i.e. due to finite precision arithmetic) and the accuracy of the solver used to obtain u^h in Eqn. (5). In all the examples we considered in this course, the solver employed by Matlab is an exact *LU* factorisation - which for the purpose of this class we will regard as having zero numerical error.

3.2 Expectations

The FE method benefits from having a very solid and rigorous mathematical background. This mathematical foundation provides many results regarding the errors we can expect from finite element calculations. We will not derive the errors estimates for the discretisation errors here, but rather state them and leave the interested reader to discover the world of finite element analysis in their own time.

In the following error estimates the coefficients C_i are constants which are independent of the grid resolution h, and the polynomial order of the basis function p. Note that a linear element implies $p = 1$, quadratic implies $p = 2$, cubic implies $p = 3$, etc. Each error estimate is associated with a different constant C_i. In general, the constants C_i cannot be derived analytically as they are problem dependent (e.g. there depend on the domain geometry, boundary conditions, coefficients in the PDE), thus if desired, they must be computed from numerical experiments. In practical computations we are usually less interested in the specific value of C_i and are primarily concerned with only the order of accuracy of the method.

STEADY STATE DIFFUSION (1D,2D,3D)

In the exercise considered in previous lessons, we used linear-1D (bilinear-2D, trilinear-3D) elements to solve the steady diffusion equation. In this case, the polynomial order of the shape function p, was $p = 1$. Using this element, the errors expected are given by

$$\int_\Omega \left|T - T^h\right| \, dV \leq C_1 h,$$

in the L_1 norm and

$$\left[\int_\Omega \left|T - T^h\right|^2 \, dV\right]^{1/2} \leq C_2 h^2,$$

in the L_2 norm. In general, for a shape function of order polynomial p, we expect the following:

$$\int_\Omega \left|T - T^h\right| \, dV \leq C_3 h^p, \tag{9}$$

$$\left[\int_\Omega \left|T - T^h\right|^2 \, dV\right]^{1/2} \leq C_4 h^{p+1}. \tag{10}$$

ELASTICITY (2D)

$$\int_\Omega \left|u - u^h\right| + \left|v - v^h\right| \, dV \leq C_5 h^p \tag{11}$$

$$\left[\int_\Omega \left|u - u^h\right|^2 + \left|v - v^h\right|^2 \, dV\right]^{1/2} \leq C_6 h^{p+1} \tag{12}$$

STOKES FLOW - MIXED METHOD (2D)

For convenience we will define the error of each velocity component as

$$e_u = u - u^h, e_v = v - v^h,$$

then the expected errors for pressure are given by

$$\left[\int_\Omega \left|p - p^h\right|^2 \, dV\right]^{1/2} \leq C_7 h^{q+1} \tag{13}$$

and for the velocity we have

$$\left[\int_\Omega \left|e_u\right|^2 + \left|e_v\right|^2 \, dV\right]^{1/2} \leq C_8 h^{p+1}. \tag{14}$$

$$\left[\int_\Omega \left|\frac{\partial e_u}{\partial x}\right|^2 + \left|\frac{\partial e_u}{\partial y}\right|^2 + \left|\frac{\partial e_v}{\partial x}\right|^2 + \left|\frac{\partial e_v}{\partial y}\right|^2 dV\right]^{1/2} \leq C_9 h^p \qquad (15)$$

Note that for the mixed method (used when solving Stokes flow), the pressure basis function usually employs a different order polynomial to the velocity basis, which here we denote via q.

Caveats

The mathematical analysis performed to obtain the above error estimates required numerous assumptions to be made. The assumptions include: homogenous material properties (diffusivity, viscosity, density, etc.); that the model domain Ω can be completely resolved by the element with zero discretisation error (i.e. the domain Ω is a rectangle and a structured grid of quadrilateral elements was used to discretise it); there is zero error associated with the boundary conditions; there is zero error associated with the matrix solver.

Thus, we emphasize very strongly, that the results above are by no means completely universal. Whilst some of the assumptions can be relaxed and the error estimates above will remain valid - however, violating these assumptions can sometimes result in completely different (usually lower) error estimates.

4 Measuring the Order of Accuracy

To keep the discussion general, let's assume that the problem of interest can be described by the abstract PDE

$$\mathcal{L}u = f,$$

which is valid over the domain of interest Ω. For this problem, we will consider that u is subject to the following Dirichlet boundary condition

$$u(\mathbf{x}) = g(\mathbf{x}) \qquad \text{for } \mathbf{x} \in \partial\Omega,$$

where the boundary of Ω is denoted by $\partial\Omega$ and $g(\mathbf{x})$ is a prescribed function (i.e. it's known). Furthermore we will assume that given f, g and the differential operator $\mathcal{L}$, we have (by some means) obtained a function u which satisfies the boundary and the PDE above, i.e. we have a solution to the problem of interest.

The procedure to measure the order of accuracy of the discretisation error can be summarised as follows:

1. Define a mesh on your domain Ω^h, with a mesh resolution of h;
2. Discretise the differential operator $\mathcal{L}$ and f, and solve for u^h;
3. Compute a global error measure, say $E(\Omega^h)_{L_2}$ for example;
4. Generate a new mesh, $\Omega^{h/2}$ by sub-dividing all the elements within the original mesh Ω^h;
5. Repeat the steps above.

If the sequence above is repeated for say 4-8 different meshes, each with successively higher resolution, the results of E (the error measured in some norm) versus h should be plotted on a $\log_{10} - \log_{10}$ graph. If the meshes used in the study are suitably fine and resolve the analytic solution well, then the plotted data in log-log space should lie along a straight line. Performing a linear regression should result in a correlation coefficient very close of 1.0. The gradient of the this log-log plot will define the order of accuracy of our numerical solution.

To understand this result, recall that the error one a given mesh domain Ω^h is assumed to vary with h according to

$$E(\Omega^h) \approx Ch^k, \tag{16}$$

where C is a constant independent of h. C is considered a function of the model problem, i.e. it relates to the shape of the domain used, the choice of boundary conditions and material properties used in the PDE. On a finer mesh, $\Omega^{h/r}$, we have

$$E(\Omega^{h/r}) \approx C\left(\frac{h}{r}\right)^k, \tag{17}$$

where r is a refinement factor. For simplicity in the steps 1-5 above we considered a refinement factor of $r = 2$. Combining Eqns. (16) & (17) we have

$$\frac{E(\Omega_1)}{E(\Omega_2)} = r^k \tag{18}$$

and now taking the $\log_{10}$ of both sides yields an expression of the order of accuracy k:

$$k = \log_{10}\left(\frac{E(\Omega_1)}{E(\Omega_2)}\right) . \left[\log_{10}(r)\right]^{-1}. \tag{19}$$

Note that we are most interested in the behaviour of the error in the asymptotic limit, $h \to 0$. If your initial mesh was too coarse and failed to resolve the analytic solution, the computed errors may not lie along a straight line. In this case, one should consider only using the errors associated with the finest meshes considered.

4.1 Asymptotics

Measuring the order of accuracy k of a method only provides information concerning how the error changes (ideally descreases) as the mesh resolution h is decreased. For example, if a method is second order accurate $\mathcal{O}(h^2)$ (in some norm), ie. $k = 2$, then we can expect that the error (measure in the appropriate norm) obtained on a grid with resolution h, will reduce be a factor of four if we solve then solve on grid with resolution $h/2$. *The order of accuracy tells us nothing about the absolute value of the error.* Another way of stating this, is the order of accuracy gives us information about the slope of the $\log(E) - \log(h)$ graph, but doesn't provide us with the offset, i.e. the constant C in Eqn. (16) is unknown.

In the asymptotic limit of $h \to 0$, then a fourth order method will always be more accurate than a second order method. However, in practice we are often far from this asymptotic limit, thus the question of which method is better cannot be answered with only knowing the order of accuracy. This point is illustrated

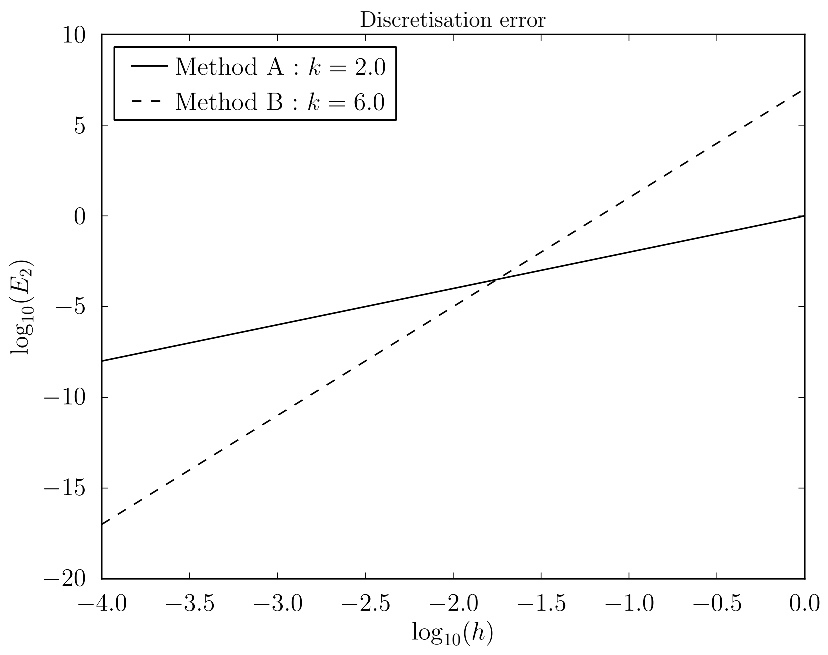

Figure 1: Discretisation errors for two methods with an order of accuracy of 6 (Method B) and 2 (Method A). Note the cross over point at $h \approx 1/56$

in Fig. 1. Pure analysis *may* provide upper bounds on what the constant C might be (under quite strict assumptions), but for all practical purposes the constant C must be determined via numerical experiments.

5 Analytic and Manufactured Solutions

In all the previous sections it was taken for granted that we also had an analytic solution u which we could use to measure the error within our numerical solution, u^h. The classical approach to obtain an analytic solution is as follows. Given the PDE of interest, $\mathcal{L}u = f$:

1. Choose a simple domain Ω (i.e. a box/cube);
2. Define the right hand side f;
3. Select your Dirichlet or Neumann boundary conditions;
4. Use all your skills and find a function u which satisfies the PDE and the boundary conditions chosen.

In general, analytic solutions for multi-dimensional problems can be difficult to obtain. The complexity of the obtaining analytic solutions continues to increase if we wish to consider coefficients (i.e. diffusivity, viscosity) which vary as a function of space, or if we were to consider a domain which was no longer naturally defined in the chosen coordinate system (i.e. a rectangle/cube in x, y, z). Up to this point, we have only consider *linear* problems. The problem is compounded if one seeks an analytic solutions for non-linear problems. Frequently researchers have to resort to using analytic solutions for 1D, homogenous coefficient problems to verify their numerical model (e.g. Poiseuille flow). Using the solution from 1D Poiseuille flow is hardly appropriate to verify a code designed to solve 3D, variable viscosity Stokes flow. An alternative approach must be employed - this approach is called the *Method of Manufactured Solutions (MMS).*

The basic idea to constructed manufactured solutions is completely general and extremely versatile. The concepts can be applied to arbitrarily complex PDE's (linear, non-linear, time dependent) and to arbitrarily complex domains and boundary conditions. The basic process of manufacturing a solution are described below. Given the PDE, $\mathcal{L}u = f$:

1. Choose the solution u_{MS};

2. Define any coefficients within $\mathcal{L}$;

3. Compute $f_{MS} = \mathcal{L}u_{MS}$ and *define the result to be your right hand side*, i.e. $f = f_{MS}$;

4. Given any domain Ω, evaluate u_{MS} or $\partial u_{MS}/\partial n$ to define any Dirichlet or Neumann boundary conditions respectively.

The procedure only requires to choose a function for the solution, and then apply the action of the differential operator $\mathcal{L}$. Whilst the algebra involved may be somewhat tedious to define the right hand side function f, we are only required to perform differentiation. The action of differentiation is easy (compared to integration) and can be readily performed by symbolic algebra packages such as Maple, Matlab (using the symbolic toolbox) Mathematica and sympy (symbolic python).

It is important to emphasize that the function chosen for u does not have to be physically meaningful. It is not important whether u "looks" like a solution you would typically obtain from your application. The purpose of manufacturing solutions is solely to ascertain that what is implemented in your code is producing discrete solutions of the quality expected by that particular numerical method.

There are some caveats to the method of manufactured solutions which should be mentioned. The function chosen for u and the coefficients in the operator $\mathcal{L}$, must be smooth and differentiable. Care should also be exercised that the constructed function f_{MS} is "well behaved", in the sense that it doesn't contain any singularities. This is easily confirmed by simply plotting the function. Furthermore, f_{MS} should be able to be resolved on the grid sequence you use when computing the order of accuracy. The coefficients should be chosen so that are physically reasonable, e.g. coefficients representing viscosity should not be negative.

5.1 Example: A linear PDE

The linear differential operator associated with the 1D diffusion equation, with a spatially variable coefficient is given by

$$\mathcal{L}u := \frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial}{\partial x}\right)T. \tag{20}$$

First I choose the coefficient for the diffusivity to be

$$\kappa(x) = (4\tanh(10x) + 6)\exp(0.8x), \tag{21}$$

and the temperature to be

$$T_{MS}(x) = 8\sin(x)x^4 + 20. \tag{22}$$

Using sympy (see `mms-linear-Diffusion1D.py`), I obtain the following right hand side

$$\begin{aligned} f_{MS} &= \frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial}{\partial x}\right)T_{MS} \\ &= [6 + 4\tanh(10x)][-8x^4\sin(x) + 64x^3\cos(x) + 96x^2\sin(x)]\exp(0.8x) \\ &+ [40 - 40(\tanh(10x)^2)][8x^4\cos(x) + 32x^3\sin(x)]\exp(0.8x) \\ &+ 0.8[6 + 4\tanh(10x)][8x^4\cos(x) + 32x^3\sin(x)]\exp(0.8x) \end{aligned} \tag{23}$$

The manufactured functions $\kappa(x)$ and u_{MS}, together with f_{MS} are shown in Fig. 2. This example is demonstrated in the script `ML_FEM_1D_DiffusionVariableCoeff_MMS.m` and the order of accuracy is determined using the resulting output. The output is processed in the script `ML_FEM_OrderOfAccuracy.m`.

6 Exercises

1. Verify your 2D steady state, diffusion code which uses Q_1 (bilinear) elements. Choose a smooth, differentiable function for u_{MS} for the temperature. Use (i) a mesh of uniform quadrilateral elements, (ii) constant thermal diffusivity $\kappa = 13.4$ and (iii) use the manufactured solution to specify Dirichlet boundary conditions everywhere. Use the global error measure in Eqn. (10). Follow the programming style in the example script `ML_FEM_1D_DiffusionVariableCoeff_MMS.m`. One important difference to note is that the example script uses 1-point Gauss quadrature to evaluate the error. In this example, you should use the 2×2 Gauss quadrature rule to evaluate Eqn. (10). Follow steps 1-5 in Section 4 and measure the order of accuracy of your FE solution by modify the script `ML_FEM_OrderOfAccuracy.m`. Deform the grid so that the elements are no longer uniform quadrilaterals and measure the order of accuracy. Does the order of accuracy change?

2. Verify your 2D elasticity which uses Q_2 (biquadratic) elements. Choose a function for the displacement solution (u_{MS}, v_{MS}) and use (i) a mesh of

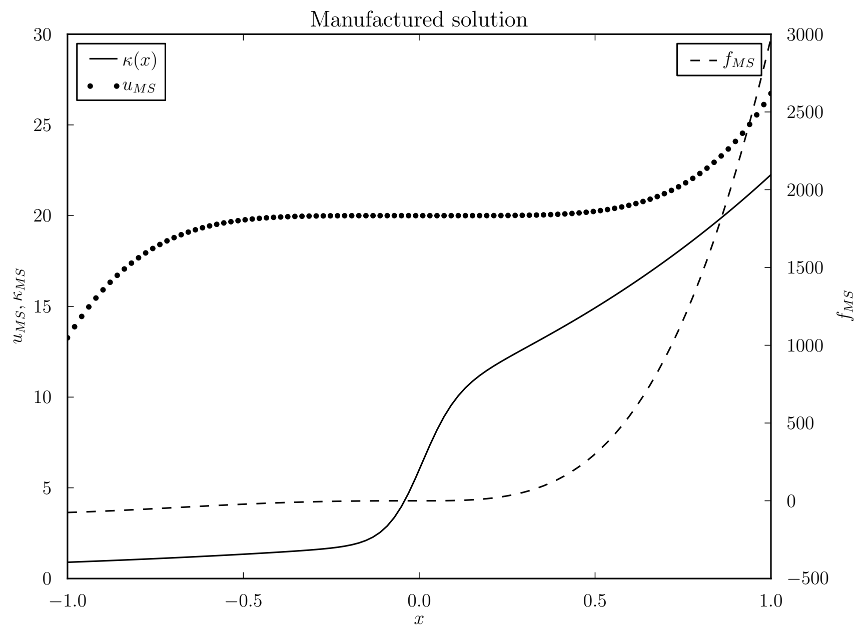

Figure 2: Manufactured solution for the 1D diffusion equation with a spatially variable diffusivity.

uniform quadrilateral elements, (ii) constant elastic parameters and (iii) use the manufactured solution to specify Dirichlet boundary conditions for both the displacement degrees of freedom (u, v). Use the error defined in Eqn. (12). Evaluate this integral using the 3×3 Gauss quadrature rule. Compute the order of accuracy of your FE solution. As an additional test, make the elasticity coefficients vary in space and see if the order of accuracy changes. Note that you will have to re-compute $\mathbf{f}_{MS}$.