

Due Monday, October 5th, 12:00 midnight

This homework is considering the analysis of 1DBVP with singularities (non-smooth solutions). A list of MatLab programs is provided together with an example problem.

Problem 1 – Boundary value problems (BVPs) with non-smooth solutions – Error analysis and higher order (quadratic and cubic) elements (MatLab)

Consider the following BVP

$$-\frac{d}{dx}\left(k(x)\frac{du}{dx}\right) = f(x), 0 \leq x \leq 1$$

where $k(x) = \frac{1}{a} + a(x - x_0)^2$ and

$$f(x) = 2\{1 + a(x - x_0)[\arctan[a(x - x_0)] + \arctan(ax_0)]\}.$$

The analytical solution to this problem (check it out!) is:

$$u_{ex}(x) = (1 - x)[\arctan a(x - x_0) + \arctan(ax_0)]$$

$$\text{and } \frac{du_{ex}}{dx} = -\{[\arctan a(x - x_0)] + [\arctan(ax_0)]\} + \frac{(1 - x)a}{1 + a^2(x - x_0)^2}$$

The boundary conditions are $u(0) = u(1) = 0$.

- Plot the analytical solution for different values of a and $x_0 = 0.5$ to realize that the function $u_{ex}(x)$ exhibits behavior which ranges from very smooth to almost discontinuous near $x = x_0$.
- Before we study this problem in more detail, modify the provided MATLAB code for 1D BVP to allow the use of quadratic and cubic elements.
- For both a 'smooth' problem (small a , e.g. $a = 5$) and a 'rough' problem (large a , e.g. $a = 50$), study the convergence rate of linear, quadratic and cubic elements. Use uniform meshes of 2, 4, 8 and 16 elements ($h = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$).

Study the convergence in both the L_2 and energy norms discussed in lecture. Plot

$\log E \equiv \log \|u - u^h\|$ versus $\log h$ and determine the experimental convergence rates for the different element types. Compare these with the theoretical values shown in class, e.g for the L_2 norm

$$\|E\|_{L_2} = \left(\int_0^1 (E(x))^2 dx\right)^{1/2} \leq C_1 h^{k+1},$$

C_1 being a constant independent of h , and for finite elements employing complete polynomials of degree k .

Similarly, we have the following result for the energy norm:

$$\|E\|_{energy} = \left(\frac{1}{2} \int_0^1 k(x) \left(\frac{dE}{dx} \right)^2 dx \right)^{1/2} \leq C_2 h^k,$$

C_2 being a constant independent of h , and for finite elements employing complete polynomials of degree k .

In your results, please normalize these computed errors with the corresponding norms of the exact solution, i.e. use the following:

$$\bar{e}_{L_2} = \frac{\left(\int_0^1 (u_{ex}(x) - u^h(x))^2 dx \right)^{1/2}}{\left(\int_0^1 (u_{ex}(x))^2 dx \right)^{1/2}}$$

and

$$\bar{e}_{energy} = \frac{\left(\frac{1}{2} \int_0^1 k(x) \left(\frac{du_{ex}(x)}{dx} - \frac{du^h(x)}{dx} \right)^2 dx \right)^{1/2}}{\left(\frac{1}{2} \int_0^1 k(x) \left(\frac{du_{ex}(x)}{dx} \right)^2 dx \right)^{1/2}}$$

The above theoretical error estimates are called asymptotic since they exhibit increasing accuracy as $h \rightarrow 0$. How small does h in your error plots need to be to achieve asymptotic convergence?

Problem 2 – Boundary value problems (BVPs) with non-smooth solutions – Adaptive finite element method (MatLab)

From Problem 1, it is seen that when the solution exhibits non-smooth behavior, the convergence of the solution is significantly reduced. A possible remedy is to adaptively refine the finite element mesh around the singular region.

Basic Ideas of Adaptive Finite Element Algorithms:

Suppose we can compute the exact error e_K in element K . If the overall target error for the computation is γ , we want that $\sum_K e_K \leq \gamma$. Having a finite element mesh, we would like to construct a new mesh, with the minimal number of nodes, such that $\sum_K e_K$ meets the

target error. This is actually a constrained nonlinear optimization problem. To facilitate its solution, we can introduce an iteration over successively finer grids. First, we need to *estimate* or bound the true error. The error bound in such estimates normally has the form $\varepsilon = \sqrt{\sum_K \varepsilon_K^2}$, where ε_K^2 is the contribution from element K . The estimates are only valid as the sum, but it is common to use the local components ε_K^2 of the sum as indicators for the refinement of individual elements. That means that we locally employ a bound of the form $e_K^2 \leq \varepsilon_K^2$. Of course, ε_K should be something that can be easily computed.

We let T^j be the mesh in the iteration j , that is, after j refinements of the initial mesh. Furthermore, let $m(T^j)$ be the number of element in T^j . We can then apply the values of ε_K computed in the current mesh T^j to select the elements to be refined. Applying a mesh refinement algorithm yields a new mesh T^{j+1} . The selection of elements to be refined is often based on the principle that the error should be uniformly distributed throughout the mesh. Thus, we aim at having $\varepsilon_K^2 \leq \gamma^2 / m_{opt}$, where m_{opt} is the number of element in the final (optimal) mesh. A natural consequence is that **an element K is marked for refinement if $\varepsilon_K^2 > \gamma^2 / m_{opt}$** . The iteration is stopped when the total estimated error ε is less than the target error γ , or when the number of refinements exceeds a prescribed maximum level. For practical computations, we often choose $\gamma = \eta \|u_h\|_{energy}$, where η is given tolerance for the relative error in the global energy norm, if the energy norm is used for the estimate. Here $\|u_h\|_{energy}$ is the energy norm of the solution and η can be seen as the tolerance in the normalized energy-norm error of the solution.

Adaptive finite element algorithm:

Choose initial mesh T^0

Compute u_h using the current mesh T^0 .

$j = 1$.

While $j \leq j_{max}$

$j = j + 1$

Compute estimator ε_K in each element

If the total error $\varepsilon = \sqrt{\sum_K \varepsilon_K^2} > \eta \|u_h\|_e$ then

Split the elements K for which $\varepsilon_K > \eta \|u_h\|_e / \sqrt{m(T^j)}$ **equally** into two smaller

(e.g. equal length) elements

Compute u_h using the new mesh T^{j+1}

Endwhile

Here the energy norm can be computed in the following way:

$$\|u_h\|_e = \left(\int_0^1 \nabla \mathbf{u}_h k(x) \nabla \mathbf{u}_h dx \right)^{\frac{1}{2}} = (\mathbf{u}_h^T \mathbf{K} \mathbf{u}_h)^{\frac{1}{2}}$$

where \mathbf{K} is the stiffness matrix.

In this problem, ε_K is chosen to be the **ZZ (Zienkiewicz-Zhu) estimator**, which is a popular indicator for low-order elements. It is defined as

$$\varepsilon_K = \left(\int_{\Omega^k} (\mathbf{q}^* - \mathbf{q}_h) k(x) (\mathbf{q}^* - \mathbf{q}_h) dx \right)^{\frac{1}{2}}$$

where $q_h = \frac{du_h}{dx}$. A very simple choice of \mathbf{q}^* is the smoothed version of q_h at the nodal points, using Galerkin, least-squares, or L_2 projection methods. A MatLab file *makeGradient.m* is provided to compute the \mathbf{q}^* given the solution vector \mathbf{u} .

Implement the Adaptive finite element algorithm with linear finite elements by modifying the 1DBVP MatLab code. Resolve the problem 1 with $a=50$ and three different $x_0 = 0.2, 0.5, 0.8$. Give the final errors comparing with analytical solution and plot the adaptive mesh. Verify that the obtained mesh is refined near x_0 . Compare your solutions with that of uniform mesh with the same number of elements in the final mesh. Consider $\eta = 5\%$ and $\eta = 1\%$ for each case to check the convergence of the algorithm. The initial mesh consists of 2 elements and the maximum number of refinement is 15.

Hint : A basic structure of the code is provided in the folder 1DBVP_adaptive. You need to finish the function ErrorEstimator.m. Make sure you understand it before you start implementation. If you like, you could still modify your own code directly on the original 1DBVP code.