Dr A. Cookson
2019/2020

Department of Mechanical Engineering
University of Bath

ME40064: System Modelling & Simulation
ME50344: Engineering Systems Simulation

Tutorial 6: Matrices for Time Integration & Error Analysis

Part A: Matrices for Time Integration

1. Write a Matlab function to calculate the local mass element matrix that multiplies the time derivative, and which is defined by:

$$Int_{mn} = \int_{-1}^{1} \psi_n \psi_m J d\xi$$

If you wish you can first calculate it performing the integral by hand, as was done in Tutorial 3 for the almost identical reaction matrix or you can move straight on to constructing this local element matrix using Gaussian Quadrature to perform the integral – following the procedure used in Tutorial 5 – Part B.Q3. With this function, and those from the previous tutorials, you now have all of the local element matrix building blocks needed to solve the transient diffusion-reaction equation.

Part B: Error Analysis

1. Write a MATLAB script to compute the L_2 norm (using Gaussian quadrature) of the error between the polynomial y = 5x^2 and a linear representation of this function on a finite element mesh with 4 elements, in the range of x=[0,1]. Note that in this example, the nodal values are exact, so all the error exists between the nodes.

2. Now run your code for Q1 but on a mesh with 1, 2, 4, 8, 16 and 32 elements, calculating the resulting element length for each of these cases.

   Represent the data as a convergence plot, using the `loglog(X,Y)` command, where *Y* is a vector containing the L_2 norm of the errors and *X* is a vector containing the corresponding mesh element sizes, *1/h*. Note by plotting the error against *log(1/h)* the slope of the line is now negative, in contrast to the example given in the lecture. However, this is in line with the convention used in the literature for this topic and it also better illustrates the fact that the error is decreasing as mesh resolution is increasing.

Calculate the gradient of your plot to check the convergence rate.

Optional Challenge Questions

3. Generalise the code written to answer Q1 so that it can take in a Matlab function - which defines the analytical solution - as an argument, along with arguments that represent the finite element representation of the data. The relevant syntax for doing this is to write:

   ```
   fun = @(x)sin(x);
   ```

   where the `@(x)` indicates that this is a function of x. The function name fun can then be passed as an argument into another function, in just the same way as a variable is passed in, and then called within that function, i.e. `fun(2)`. For more information, read this documentation page:

https://uk.mathworks.com/help/matlab/matlab_prog/pass-a-function-to-another-function.html

4. Modify the Gaussian quadrature structure that was used in the previous tutorial to include N=4 and N=5 order GQ schemes. Test these schemes with a higher-order polynomial of your choice using your code from Q3. The Gauss weights and Gauss points for N=4 are:

$$\xi_1, \xi_2 = \pm\sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}, \quad w_1, w_2 = \frac{18+\sqrt{30}}{36}$$

$$\xi_3, \xi_4 = \pm\sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}, \quad w_3, w_4 = \frac{18-\sqrt{30}}{36}$$

The Gauss weights and Gauss points for N=5 are:

$$\xi_1 = 0$$

$$\xi_2, \xi_3 = \pm\frac{1}{3}\sqrt{5 - 2\sqrt{\frac{10}{7}}}$$

$$\xi_4, \xi_5 = \pm\frac{1}{3}\sqrt{5 + 2\sqrt{\frac{10}{7}}}$$

$$w_1 = \frac{128}{225}$$

$$w_2, w_3 = \frac{322 + 13\sqrt{70}}{900}$$

$$w_4, w_5 = \frac{322 - 13\sqrt{70}}{900}$$