MA378 – Numerical Analysis II

Lab 3: Gaussian Quadrature

Goal: Derive Gaussian Quadrature rules using Maple and implement them using Matlab

Quadrature Rules are methods of approximating definite integrals, e.g., $\int_a^b f(x)dx$. The $(n+1)$-point Gaussian rule is of the form

$$G_n(f) := w_0 f(x_0) + w_1 f(q_1) + \cdots + w_n f(x_n).$$

where the quadrature points, x_i, and the weights, w_i, are chosen is some "optimal" way.

There are different approaches to this, all leading to the same values. Two of them are

(i) Choose the x_i and the w_i so that rule has precision $2n+1$. This can be done by the *method of undetermined coefficients*, but requires us to solve a system of nonlinear equations. For $n \geq 2$ there is a better alternative.

(ii)
- Define the inner product $(f,g) := \int_a^b f(x)g(x)dx$.
- Construct a sequence of polynomials $\{\tilde{p}_k(x)\}_{k=0}^{\infty}$ where each $\tilde{p}_n(x)$ is monic of degree n, and orthogonal to all polynomials of degree less than k with respect to the above inner product.
- Take the x_i to be the $n+1$ zeros of $\tilde{p}_{n+1}(x)$.
- Set the *quadrature weights* $w_0, w_1, \ldots, w_n$ to be so that

$$w_i = \int_a^b L_i(x)dx = \int_a^b \prod_{k=0, k\neq i}^{n} \frac{x - x_k}{x_i - x_k} dx.$$

The task of finding the $\tilde{p}_{n+1}$, its zeros, and integrating the corresponding $L_i(x)$ is tedious but can easily be done using a symbolic mathematics package, such as Maple.

1. The Maple Program `Gaussian1.mw` uses the *Method of Undetermined Coefficients* to derive the rule

$$G_1(f) = w_0 f(x_0) + w_1 f(x_1) \approx \int_{-1}^{1} f(x)dx,$$

based on the method of undetermined coefficients. Try adapting this Maple code to derive a 3-point Gaussian Quadrature Rule (See Exercise *3.5.3*).

2. The Maple Program `Monic1.mw` uses the Gram-Schmidt method to compute $\{\tilde{p}_0, \tilde{p}_1, \tilde{p}_2\}$. Then the zeros of p_2 are used as quadrature points. Again, try adapting this code to derive the 3-point rule.

3. Download the Matlab file `NewtonCotes.m`. This implements the 2, 3 and 4-point Newton-Cotes methods to estimate $\int_{-1}^{1} \ln(2+x)dx$. Verify that you get the same results has given below.

   Write a similar program to implement the Gaussian rules $G_1(\cdot)$ and $G_2(\cdot)$. Compare their accuracy with the corresponding Newton-Cotes Rule.

   If you have the time and inclination, try deriving and implementing $Q_4(\cdot)$ and $G_3(\cdot)$ rules.

Method		Quadrature Points (x_0, x_1, etc)	Weights (w_0, w_1, etc)	Estimate	Error
Newton-Cotes	$Q_1(\cdot)$	$-1, 1$	$1, 1$	1.098612	1.972×10^{-1}
Newton-Cotes	$Q_2(\cdot)$	$-1, 0, 1,$	$\frac{1}{3}, \frac{4}{3}, \frac{1}{3}$	1.290400	5.437×10^{-3}
Newton-Cotes	$Q_3(\cdot)$	$-1, -\frac{1}{3}, \frac{1}{3}, 1$	$\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4}$	1.293246	2.591×10^{-3}
Newton-Cotes	$Q_4(\cdot)$				
Gaussian	$G_1(\cdot)$				
Gaussian	$G_2(\cdot)$				
Gaussian	$G_3(\cdot)$				

When you are done, upload your Matlab code that implements G_1 and G_2 to the "`labs`" section of the MA378 Blackboard module. Make sure it includes your name and ID number.

Deadline: 5pm, Monday, 20 March, 2017.