Math 6316, Spring 2016
Due 28 March 2016

Homework 5 – Orthogonal Polynomials

1. Recall the definition of the Chebyshev polynomials: for $x \in [-1, 1]$,

$$T_k(x) = \cos\left(k \arccos(x)\right), \quad k = 0, 1, \ldots.$$

Derive the 3-term recurrence relation

$$T_0(x) = 1, \qquad T_1(x) = x, \qquad T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad k = 1, 2, \ldots.$$

Show all relevant steps in your derivation. *Hint: set* $x = \cos(\theta)$ *for* $\theta \in [0, \pi]$.

2. Derive the nodes $\{x_k\}_{k=0}^2$ and weights $\{\alpha_k\}_{k=0}^2$ for the Gauss quadrature formula where one node is required to equal 1, i.e.

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \;\approx\; \alpha_0 f(x_0) + \alpha_1 f(x_1) + \alpha_2 f(1).$$

Show all relevant steps in your derivation. What is the degree of precision for your quadrature formula? Based on this, when used in a composite quadrature approach for subintervals of size H, what is the expected rate of convergence (i.e. r such that error is $\mathcal{O}(H^r)$)? *Hint: follow a similar approach to the one laid out for Gauss-Lobatto quadrature in equation (10.17) in the book.*

3. Matlab: Using your result from problem 2, construct a function

```
function [Imn, nf] = composite_quad(f, a, b, m)
```

that implements a composite version of your 3-node quadrature formula. The inputs to this function are: `f` is the integrand function handle or name, `a` and `b` define the integration interval $[a, b]$, and `m` is the number of subintervals to use in the composite method.

The outputs from this function are: `Imn` is the approximation to $\int_a^b f(x)\,\mathrm{d}x$, and `nf` is the number of calls to the integrand function `f`.

Write a script named `hw5.m` to test this quadrature function on the problem

$$\int_0^{\pi} e^{2x} \cos(3x)\,\mathrm{d}x.$$

Compare your method against the composite Gauss3 and Simpson methods, using $m_j = 4 * 2^j$ for $j = 0, \ldots, 6$ subintervals. For each test compute the absolute error as compared with the analytical solution. For $j = 1$ and higher, output an estimate of the convergence rate for each method by comparing two consecutive m and error values e, i.e.

$$r_j = -\frac{\log(e_j) - \log(e_{j-1})}{\log(m_j) - \log(m_{j-1})}.$$

Note: again, I will call your code from a custom testing script, so ***do not deviate from the specified function arguments****.*