HW 6

Due at 11:59 by CMS on Monday, April 18

Remember that you may (and should!) talk about the problems amongst yourselves, or discuss them with me or the TA, providing attribution for any good ideas you might get – but your final write-up should be your own.

1: Testing Simpson. Implement the following routine:

```
% I = hw6simpson(f, a, b, n)
%
% Integrate f (passed as a MATLAB function handle) from a to b
% using n-panel Simpson quadrature.

function I = hw6simpson(f, a, b, n)
```

In addition, write a testing script to check the following features:

- The interval specified by $[a, b]$ is used. You should test that your code works properly for the case $a = b$, and also for the case where $b < a$ (using the convention that $\int_a^b f(x)\,dx = -\int_b^a f(x)\,dx$.
- The quadrature rule has degree 3 (i.e. cubics are integrated exactly but quartics need not be).
- The function has the desired order of convergence on the test integrand e^x for the interval $[0, 1]$. You should do this by repeatedly doubling n and showing that the error decreases appropriately.

Your tester should output a diagnostic failure method if `hw6simpson` is incorrect. I will be checking your test script by making sure that it reports success for a correct implementation of Simpson's rule and reports failure for some incorrectly-implemented variants of Simpsons rule (including versions that correctly estimate the integral, but do not have the right order of convergence).

2: Legendre revisited. In lecture 20, we described the Legendre polynomials, and gave a recurrence for computing them. For any sufficiently nice (at least square integrable) function $f(x)$ on $[-1, 1]$, we can write an expansion in terms of the Legendre polynomials:

$$f(x) = \sum_{j=0}^{\infty} c_j P_j(x),$$

Note that

$$\int_{-1}^{1} f(x)P_k(x)\,dx = \sum_{j=0}^{\infty} c_j \int_{-1}^{1} P_j(x)P_k(x)\,dx = \frac{2c_j}{2j+1},$$

so we can compute the coefficients by integration.

Write a function that estimates $c_0, \ldots, c_{n-1}$ using n-point Gauss quadrature, and a second function that evaluates

$$\hat{f}(x) = \sum_{j=0}^{n-1} c_j P_j(x)$$

at given nodes in x. Your functions should have the form

function c = hw6expansion(f,n)
function fhatx = hw6eval(c,x)

Here `f` is a function handle, `n` is the number of coefficients, `c` is the coefficient vector, and `x` is a vector of points where the function should be evaluated. Explain why this construction of $\hat{f}(x)$ is equivalent to polynomial interpolation through the nodes used in the n-point Gauss quadrature rule.

You may use `gaussq.m` (posted on CMS) to compute the necessary Gauss quadrature nodes and weights.