MAT128a: Numerical Analysis, Fall 2015
Programming Project Two
Due: November 4, 2015

In this project, you will write three MATLAB functions which perform tasks involving orthogonal polynomials. Below, I describe the operation of each function and provide a template indicating its syntax. Follow these templates exactly — we will grade the project by calling your functions and testing to see that they perform as expected. The templates are also available on the course website. There are also four "test scripts" available at the course website. I provide them in order to help you ensure your project runs properly.

Each of the functions you write should be placed in an ".m" file whose name is the same as the name of the function. You will submit your project by sending an email to the following address:

`mat128a_fall2015@math.ucdavis.edu`

Please send one email with three attachments, one for each of the ".m" files which comprise this project. You will get a reply with your score within two weeks of submission (probably much sooner). Please try to avoid making multiple submissions — wait to submit your project until you are completely satisfied with it. If you do submit more than once, only the last submission will be graded. Moreover, only submissions *received* before 11:59 PM on the due date will be considered.

Project description

1. Write a MATLAB function called "legendre" which takes as input a nonnegative integer n and a collection of real numbers $x_1, \ldots, x_n$ in the interval $(-1, 1)$ and returns as output an array contianing the values of the Legendre polynomial of degree n at the points $x_1, \ldots, x_n$.

2. Write a MATLAB function called "legecoefs" which takes an input a function defined on $[-1, 1]$ and a nonnegative integer n and returns as output an array containing the $(n + 1)$ values $a_0, a_1, \ldots, a_n$ defined via

$$a_k = \int_{-1}^{1} P_k(x) f(x) dx, \tag{1}$$

where P_k is the Legendre polynomial of order k.

Use the MATLAB builtin function "integral" in order to evaluate the integrals in (1). The following code returns the definite integral of x^2 over the interval $[-1, 1]$:

```
fun = @(x) x.^2;
integral(fun,-1,1)
```

Note that the code

```
fun = @(x) x^2;
integral(fun,-1,1)
```

fails. This is because the input function "fun" must be "vectorized." That is, rather than taking as input a single real number x and returning the value of $f(x)$, the input function must take as input an array and return the value of f at each point in the array. The MATLAB operator "^" is not vectorized ; however, the version ".^" is. This is why the second code snipet above works and the first does not. As a further example:

```
fun = @(x) x.^2 .* exp(-x.^2) .* cos(x)
integral(fun,-1,1)
```

computes the value of the definite integral

$$\int_{-1}^{1} x^2 \exp(-x^2) \cos(x)\, dx. \tag{2}$$

3. Write a MATLAB functions called "legeeval" which takes as input the $n+1$ coefficients $a_0, a_1, \ldots, a_n$ in the expansion

$$p(x) = a_0 P_0(x) + a_1 P_1(x) + \cdots + a_n P_n(x), \tag{3}$$

where P_k is the Legendre polynomial of order k, and a collection of real number $x_1, \ldots, x_m$ in the interval $(-1, 1)$ and returns an array containing the values of the function p defined via (3) at $x_1, \ldots, x_m$.

```
function vals = legendre(n,xs)
% functions vals = legendre(n,xs)
%
% Return the values of the (n+1) Legendre polynomial of degree
% n at a collection of points x_1,...,x_m in (-1,1).
%
% Input parameters:
%   n - the order of the polynomial to evaluate
%   xs - a vector of length m whose jth entry specifies the point x_j
%
% Output parameters:
%   vals - a vector of length m whose jth entry specifies the value
%          of the Legendre polynomial of degree n at the point x_j
%

vals = zeros(size(xs));

end
```

```
function as = legecoefs(n,fun)
%  function coefs = legecoefs(n,funs)
%
%  Return the (n+1) coefficients a_0, a_1, ..., a_n such that
%
%    p(x) = a_0 P_0(x) + a_1 P_1(x) + ... + a_n P_n(x),
%
%  where P_j denotes the Legendre polynomial of degree j, is the
%  unique polynomial of degree n which minimizes the quanitity
%
%          1
%     \int      |p(x)-f(x)|^2 dx.
%         -1
%
%  Here, f is a user-specified external function.  Of course,
%  a_j is equal to
%
%                                1
%     a_j = 2/(2j+1)     \int   P_j(x) f(x) dx
%                               -1
%
%
%  Input parameters:
%    n - an integer specifying the  order of the approximating polynomial
%    fun - an external function which takes as input a vector of doubles
%      xs and returns the vector of doubles ys obtained by evaluating
%      f at each of the entries of the array xs; for example:
%
%              fun = @(x) x.^2
%
%              fun([1 2]) returns [2 4]
%
%
%  Output parameters:
%    as - a vector of length n+1 whose j_th entry is the value
%         of the coefficient a_(j-1)
%
%
%

as = zeros(n+1,1);
end
```

```
function vals = legeeval(n,as,xs)
% function vals = legeeval(n,as,xs)
%
% Evaluate a (n+1)-term Legendre expansion
%
%   p(x) = a_0 P_0(x) + a_1 P_1(x) + ... + a_n P_n(x)
%
%  at a collection of user-specified points x_1,...,x_m in (-1,1).
%
%  Input parameters:
%
%    n - the number of terms in the expansion
%    as - a vector of length n+1 whose j^th entry specifies the
%        coefficient a_(j-1)
%    xs - a vector of length m whose j^th entry is the point x_j
%
%
%  Output parameters:
%
%    vals - a vector of length m whose j^th entry is the value
%          of p at the point x_j
%

vals = zeros(size(xs));

end
```