March 17, 2017

Lecture 21
Numerical integration: Newton-Cotes quadrature rules, Part II

By the end of this lecture, students will be able to

- implement Newton-Cotes quadrature rules in Matlab
- understand its approximation properties.

1 Implementation

For convenience in subsequent computations, we package t he Newton-Cotes weight vectors in the following function:

```
function w = NCWeights(m)
% w = NCWeights(m)
%
% w is a column m-vector consisting of the weights for the m-point
% Newton-Cotes rule on the unit interval [0,1].
% m is an integer that satisfies 2 ⩽ m ⩽ 8.

if m==2
        w=[1 1]'/2;
elseif m==3
        w=[1 4 1]'/6;
elseif m==4
        w=[1 3 3 1]'/8;
        :
end
```

Notice that the weight vectors are symmetric about their middle in that

```
w(1:m) = w(m:1:1)
```

Turning now to the evaluation of $Q_{NC(m)}$ itself, we see from

$$Q_{NC(m)} = (b-a)\sum_{i=1}^{m} w_i f_i = (b-a)[w_1\ w_2\ \cdots\ w_m]\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$$

that it is a scaled inner product of the weight vector w and the vector of function values. Therefore, we obtain

```
function numI = QNC(f,a,b,m)
% m-point Newton-Cotes quadrature across the interval [a b].
% f is a handle that points to a function of the form f(x) where x is a
% scalar. f must be defined on [a,b] and it must return a column vector if
% x is a column vector.
% m is an integer that satisfies 2 ⩽ m ⩽ 8.
% numI is the m-point Newton-Cotes approximation of the integral of f from
% a to b.
w = NCweights(m);
x = linspace(a,b,m)';
fvals = f(x);
numI = (b-a)*(w'*fvals);
```

✍ Use m-point Newton-Cotes rules with $m = 2, 3, \cdots, 8$ to compute the integral of $f_1(x) = e^{-x}$ and $f_2(x) = e^{-20x}$ on $[0, 1]$, record the numerical error in a table.
File: `testQNC.m, QNC.m, NCWeights.m`

2 Newton-Cotes Error

How good are the Newton-Cotes rules? Since they are based on the integration of a polynomial interpolant, the answer clearly depends on the quality of the interpolant. Here is a result for Simpsons rule:

Theorem 1. *If* $f(x)$ *and its first four derivatives are continuous on* $[a, b]$*, then*

$$\left| \int_a^b f(x)dx - Q_{NC(3)} \right| \leqslant \frac{(b-a)^5}{2880} \max_{x\in[a,b]} |f^{(4)}(x)|.$$

Note that if $f(x)$ is a cubic polynomial, $f^{(4)}(x) = 0$, so Simpson's rule is exact. This is somewhat surprising because the rule is based on the integration of a *quadratic* interpolant.

In general, it can be shown that

$$\int_a^b f(x)dx = Q_{NC(m)} + c_m f^{(d+1)}(\eta) \left(\frac{b-a}{m-1} \right)^{d+2},$$

where c_m is a constant, $\eta \in [a, b]$, and

$$d = \begin{cases} m-1 & \text{if } m \text{ is even,} \\ m & \text{if } m \text{ is odd.} \end{cases}$$

The following function can be used to return this upper bound given the interval $[a, b]$, m, and the appropriate derivative bound:

```
function error = QNCError(a,b,m,M)
% The error bound for the m-point Newton-Cotes rule when applied to
% the integral from a to b of a function f(x). It is assumed that
% a<=b and 2 <= m <= 8. M is an upper bound for the (d+1)-st derivative of the
% function f(x) on [a,b] where d = m if m is odd, and m-1 if m is even.
if m==2, d=1; c = -1/12;
elseif m==3, d=3; c = -1/90;
elseif m==4, d=3; c = -3/80;
elseif m==5, d=5; c = -8/945;
elseif m==6, d=5; c = -275/12096;
elseif m==7, d=7; c = -9/1400;
elseif m==8, d=7; c = -8183/518400;
else         d=7; c = -8183/518400;
end
error = abs( c*M*((b-a)/(m-1))^(d+2));
```

✍ Use m-point Newton-Cotes rules with $m = 2, 3, \cdots, 8$ to compute the integral of $f(x) = \sin(x)$ on $[0, \pi/2]$, record the numerical error in a table, along with the error bound.

File: `testQNCError.m, QNCError.m`