Math 151A Homework #6 – due Wednesday 12/06, in class

Show all your work!

1. Composite midpoint rule

Derive the expression for the composite midpoint rule (theorem 4.6). The derivation is similar to the derivation of the composite Simpson's rule, which we did in class.

Answer:

Let $f \in C^2[a,b]$, n even, $h = (b-a)/(n+2)$ and $x_j = a + (j+1)/h$. Then we can apply the midpoint rule on each subinterval (see figure 4.9 in the book):

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{j=0}^{n/2} \int_{x_{2j-1}}^{x_{2j+1}} f(x)dx \\ &= \sum_{j=0}^{n/2} \Big[2hf(x_{2j}) + \frac{h^3}{3} f''(\xi_j)\Big] \qquad x_{2j-1} < \xi_j < x_{2j+1} \\ &= 2h \sum_{j=0}^{n/2} f(x_{2j}) + \frac{h^3}{3} \sum_{j=0}^{n/2} f''(\xi_j)\end{aligned}$$

Since $f \in C^2[a,b]$, the Extreme Value Theorem implies that f'' assumes its maximum and mininmum value in $[a,b]$. Since

$$\min_{x\in[a,b]} f''(x) \le f''(\xi_j) \le \max_{x\in[a,b]} f''(x)$$

we have

$$\big(\frac{n}{2}+1\big) \min_{x\in[a,b]} f''(x) \le \sum_{j=0}^{n/2} f''(\xi_j) \le \big(\frac{n}{2}+1\big) \max_{x\in[a,b]} f''(x)$$

and

$$\min_{x\in[a,b]} f''(x) \le \frac{2}{n+2} \sum_{j=0}^{n/2} f''(\xi_j) \le \max_{x\in[a,b]} f''(x)$$

By the Intermediate Value Theorem, we see that there exists a $\mu \in (a,b)$ such that

$$f''(\mu) = \frac{2}{n+2} \sum_{j=1}^{n/2} f''(\xi_j)$$

Thus we get

$$\begin{aligned}\int_a^b f(x)dx &= 2h \sum_{j=0}^{n/2} f(x_{2j}) + \frac{(n+2)h^3}{6} f''(\mu) \\ &= 2h \sum_{j=0}^{n/2} f(x_{2j}) + \frac{(n+2)h^2(b-a)}{6(n+2)} f''(\mu) \\ &= 2h \sum_{j=0}^{n/2} f(x_{2j}) + \frac{(b-a)}{6} h^2 f''(\mu)\end{aligned}$$

which is the Composite Midpoint rule.

2. Gaussian quadrature

According to theorem 4.7, we can approximate the integral of a function as

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} c_i f(x_i)$$

where

$$c_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j} dx$$

and $\{x_j\}$ are the roots of the nth Legendre polynomial.

Show that for $n = 1$ this gives the midpoint rule

$$\int_a^b f(x)dx = (b-a)f(\frac{b+a}{2})$$

Answer:

For $n = 1$ we have

$$\int_{-1}^{1} f(x)dx \approx c_1 f(x_1)$$

where x_1 is the root of the first Legendre polynomial $P_1(x) = x$ (which is $x_1 = 0$), and c_1 is given by

$$c_1 = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq 1}}^{1} \frac{x - x_j}{x_i - x_j} dx = \int_{-1}^{1} dx = 2$$

This yields

$$\int_{-1}^{1} f(x)dx \approx 2f(0)$$

To compute the integral

$$\int_a^b f(x)dx$$

we use the variable substitution $u = 2(x-a)/(b-a) - 1$, which means that

$$x = u\frac{b-a}{2} + \frac{b+a}{2}$$

and

$$dx = \frac{b-a}{2} du$$

So then

$$\begin{aligned}
\int_a^b f(x)dx &= \frac{b-a}{2} \int_{-1}^{1} f(\frac{b-a}{2}u + \frac{b+a}{2})du \\
&\approx \frac{b-a}{2} \cdot 2f(\frac{b-a}{2} \cdot 0 + \frac{b+a}{2}) \\
&= (b-a)f(\frac{b+a}{2})
\end{aligned}$$

3. Gaussian elimination [Computational]
Implement the Gaussian elimination with partial pivoting algorithm (algorithm 6.2) and use it to solve the following system

$$\begin{aligned} 3.03x_1 - 12.1x_2 + 14x_3 &= -119 \\ -3.03x_1 + 12.1x_2 - 7x_3 &= 120 \\ 6.11x_1 - 14.2x_2 + 21x_3 &= -139 \end{aligned}$$

Answer:
Here is the Matlab/Octave code for Gaussian elimination with partial pivoting:

```
%-----------------------------------------------------------
% Gaussian elimination with partial pivoting
% Must put augmented matrix in "A"
% outputs values x1 ... xn
%-----------------------------------------------------------

% get the size of the matrix
n = length(A) - 1;

% we don't want to actually interchange the values in
% two rows, since that is slow.  Instead we use a "pointer"
% to the rows.
for i = 1:n
    nrow(i) = i;
end

% Now do Gaussian elimintaion
for i = 1:(n-1)

    % find the maximum element in the pivot element's column
    max = abs(A(nrow(i),i));
    jmax = nrow(i);
    for j = (i+1):n
        if abs(A(nrow(j),i)) > max
            jmax = j;
            max = abs(A(nrow(j),i));
        end
    end

    % if max of the column is zero, no solution
    if max == 0
        printf("No unique solution exits.\n");
        return;
    end

    % if jmax != i, then swap rows
    if nrow(jmax) != nrow(i)
        temp = nrow(i);
        nrow(i) = nrow(jmax);
```

```
        nrow(jmax) = temp;
    end

    % Now do the actual elimination step
    for j = (i+1):(n)
        m(nrow(j),i) = A(nrow(j),i)/A(nrow(i),i);
        for k = (i+1):(n+1)
            A(nrow(j),k) = A(nrow(j),k) - m(nrow(j),i)*A(nrow(i),k);
        end

    end
end

% now backward substitution
if A(nrow(n),n) == 0
    printf("No unique solution exists.\n");
    return;
end
x(n) = A(nrow(n),n+1)/A(nrow(n),n);
for i = (n-1):-1:1
    x(i) = A(nrow(i),n+1);
    for j = (i+1):n
        x(i) = x(i) - A(nrow(i),j)*x(j);
    end
    x(i) = x(i) / A(nrow(i),i);
end
x
```

And here is the output:

```
octave:1> A = [3.03,-12.1,14,-119;-3.03,12.1,-7,120;6.11,-14.2,21,-139]
A =

     3.0300   -12.1000    14.0000  -119.0000
    -3.0300    12.1000    -7.0000   120.0000
     6.1100   -14.2000    21.0000  -139.0000

octave:2> gepp
x =

   0.00000  10.00000   0.14286
```