Numerical Computing II

Homework 18: Quadrature

Marty Fuhry

March 8, 2010

Exercise 18.2

We're looking to determine weights w_1, w_2, w_3 such that

$$\int_0^1 f(x), dx \approx f(x_1)w_1 + f(x_2)w_2 + f(x_3)w_3$$

integrates quadratic polynomials exactly. That is, we want to determine weights that solve a system of equation where each weight contributes to perfectly integrate up to a quadratic polynomial. So we need to solve the system:

$$\begin{aligned} w_1 + w_2 + w_2 &= b - a \\ x_1 w_1 + x_2 w_2 + x_3 w_3 &= \frac{1}{2}(b^2 - a^2) \\ x_1^2 w_1 + x_2^2 w_2 + x_3^2 w_3 &= \frac{1}{3}(b^3 - a^3). \end{aligned}$$

We are given $x_1 = 0, x_2 = \frac{1}{2}, x_3 = 1$ and $a = 0, b = 1$. Then, we can use

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1/2 & 1 \\ 0 & 1/4 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \end{bmatrix}$$

which corresponds to the Octave code:

```
x1 = 0;
x2 = 1/2;
x3 = 1;

a = 0;
b = 1;

V = vander([x1,x2,x3]);
V = fliplr(V)';

y = [b-a;  1/2(b^2-a^2);  1/3(b^3-a^3)];

sol = V\y;
```

This will yield the solution vector

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1/6 \\ 2/3 \\ 1/6 \end{bmatrix}.$$

This means, we can say with exactness, if $f(x)$ is a quadratic polynomial or less, then

$$\begin{aligned} \int_0^1 f(x)dx &= f(x_1)w_1 + f(x_2)w_2 + f(x_3)w_3 \\ &= \frac{1}{6}f(0) + \frac{2}{3}f(\frac{1}{2}) + \frac{1}{6}f(1) \end{aligned}$$

This will not integrate polynomials of higher degree exactly. This is because in the linear system of equations we solved, that is, the Vandermonde matrix with exactly three rows and three columns, we did not have enough data to force our weights to line up exactly with higher degree polynomials. If we had more data points, x_i, we would be able to determine weights which could integrate higher degree polynomials exactly. If we wanted to interpolate a third degree polynomial exactly, we would need a fourth weight. If we had a fourth weight, w_4, we would need a fourth node x_4 to solve for the fourth weight, and thus, we would need the function evaluated at that fourth node.

For the interval [1,2], we perform a change of variables. We define the function $x(t)$ on the interval [-h,h] where $h \in \mathbb{R}$. We say $x(-h) = 1$ and $x(h) = 2$. Then, the function $x'(t) = \frac{1}{2h}$. We also define the function $g(t)$ such that:

$$\begin{aligned} g(t) = f(x(t)) \implies \\ \int_1^2 f(x)dx &= \int_{-h}^{h} f(x(t))x'(t)dt \\ &= \int_{-h}^{h} g(t)\frac{1}{2h}dt \\ &= \frac{1}{2h}\int_{-h}^{h} g(t)dt \\ &\approx \frac{1}{2h}(w_1 g(-h) + w_2 g(0) + w_3 g(h)) \\ &\approx \frac{1}{2h}(w_1 f(1) + w_2 f(1/2) + w_3 f(2)) \end{aligned}$$

Now, remembering from our system of equations for solving $f(t) = t$ and $f(t) = t^2$,

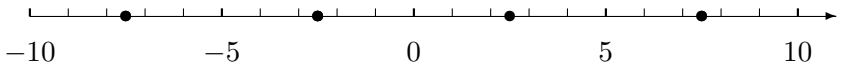
$$\begin{aligned} -w_1 h + w_3 h = 0 \\ w_1 + w_3 = \frac{2h}{3} \end{aligned}$$

We solve for $w_1 = w_3 = \frac{h}{3}$ and from $w_1 + w_2 + w_3 = 2h$, we solve $w_2 = \frac{4h}{3}$. Then,

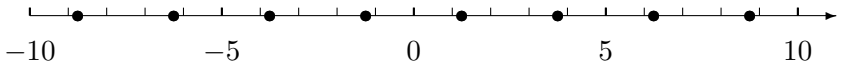
$$\begin{aligned} \int_1^2 f(x), dx &\approx \frac{1}{2h}(w_1 f(1) + w_2 f(1/2) + w_3 f(2)) \\ &\approx \frac{1}{2h}(\frac{h}{3}f(1) + \frac{4h}{3}f(1/2) + \frac{h}{3}f(2)) \\ &\approx \frac{1}{6}(f(1) + 4f(1/2) + f(2)) \end{aligned}$$

Exercise 18.4

The interval [-10,10] may be evaluated at 4 points like this using $N = 4$:

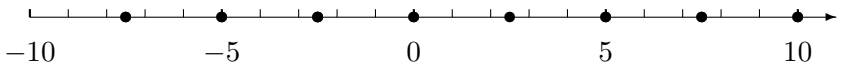


Now, a naive approach to reevaluating the function for $N = 8$ points would try something like this:



But none of the points have matched up from the second evaluation! We have to reevaluate every single point! This is terrible, and clearly not a good way to proceed.

Luckily, we can easily rectify this by simply "shifting" the points from the second evaulation to line up with the first evaulation, like this:



Now we only need to evaulate 4 more points: $f(-5), f(0), f(5), f(10)$ and reusing the old points we evaluated for the previous quadrature evaluation. In general, to double N, we need only compute N more points.

However, this method is only suitable for quadrature rules which are only concerned with endpoints. That is, a midpoint rule would NOT be able to take advantage of this optimization since the midpoints would no longer be midpoints after each evaluation. We would need to either keep every midpoint and shrink the computed integral range, or just reevaluate new midpoints for each and every new iteration.

A composite midpoint rule, then, would not be able to reuse any points from this algorithm. It would have to recompute points after each and every iteration without any reuse.

Exercise 18.5

Part (a)

We'll start by using the Taylor Expansion of $f(x)$ about the point $h/2$:

$$f(x) = f(h/2) + (x - h/2)f'(h/2) + 1/2(x - h/2)^2 f''(h/2) + \cdots$$

Then,

$$\int_0^h f(x)dx = hf(h/2) + 1/24h^3 f''(h/2) + \cdots$$

Now, we're interested in finding the error between the actual value and the computed value. Now, we compute a different value for each h, and each $T(f)$ looks like:

$$T(f) = h/2(f(0) + f(h)).$$

We solve for $f(0)$ and $f(h)$ using the Taylor Expansion.

$$\begin{aligned} f(0) &= f(h/2) + -h/2f'(h/2) + 1/2(h/2)^2 f''(h/2) + \cdots \\ f(h/2) &= f(h/2) + h/2f'(h/2) + 1/2(h/2)^2 f''(h/2) + \cdots . \end{aligned}$$

Then, when we add these together, the first order h is eliminated and we are left with:

$$h/2(f(0) + f(h)) = hf(h/2) + 1/8h^3 f''(h/2) + \cdots .$$

Part (b)

Now, we can this equation to solve for the highest order part of our error equation:

$$\begin{aligned} \int_0^h f(x)dx - T(f) &= hf(h/2) + 1/24h^3 f''(h/2) + \cdots \\ &\quad - hf(h/2) - 1/8h^3 f''(h/2) - \cdots \\ &\approx 1/24h^3 f''(h/2) - 1/8h^3 f''(h/2) \\ &\approx -1/12h^3 f''(h/2). \end{aligned}$$

So this is the error for each composite h. When we add all the errors together for the entire quadrature, we get

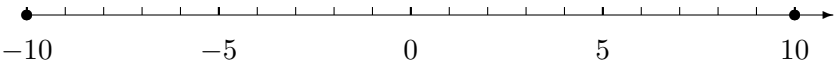
$$\begin{aligned}(N-1)*(-1/12h^3f''(h/2)) = \\ h(N-1)*(-1/12h^2f''(h/2)) = \\ (b-a)*(-1/12h^2f''(h/2)).\end{aligned}$$

This tells us that our error term is $O(h^2)$. When h is halfed, we should be able to reduce error by a magnitude $1/4$.

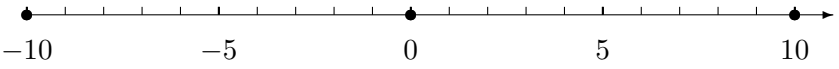
Note: I'm not sure about that negative sign. I must have checked over my equations a dozen times and didn't see anything offensive. Though, I'm not sure if the negative sign isn't supposed to be there. Maybe when we have a positive second derivative (a concave up function), we will approximate small. That is, we will get less than the appropriate value. Of course, all these error functions are only concerned with the absolute value of our error, anyway, so this really shouldn't be an issue.

Part (c)

We can finally use the awesome optimization discussed in Exercise 18.4. For our first iteration, we compute $f(a)$ and $f(b)$. Imagine we are using the compositive trapezoid quadrature rule on the interval [-10,10], as before. We first use only $f(-10)$ and $f(10)$:



Then, when we want to increase our accuracy, we divide our interval into two subintervals by using the point $f(0)$.



We now have a more accurate approximation, but we only evaluated one more point. In general, we can reuse all of our points after every iteration. That means, to compute $T_{h/2}(f)$, we need to only evaluate half of the points used by reusing all of the old points from $T_h(f)$.

Exercise 18.6

This program will calculate the quadrature using the composite midpoint rule at N distinct points.

```
% define endpoints
a = 0;
b = 1;

% 'exact' answer given by quad
exact = quad(@f,0,1);

% define N and h for 2 nodes
N = 100;
h = (b - a) / (N);

% generate x and f(x)
x = [];
y = [];
for j = 1:N
    x(j) = a + (j - .5)*h;
    y(j) = f(x(j));
end

% determine quadrature
approx = 0;
for j = 1:N
   approx = approx + 1/N * y(j);
end
```

When we run this for $N = 2$, we obtain the 'exact' value using Octave's 'quad' command, which uses Gaussian Quadrature and yields -1.1351. The following table shows the composite midpoint rule's computed values and errors for different values of N.

N	Computed Value	Error
2	-0.99030	0.21254
10	-1.10160	0.033505
100	-1.1317	0.0034049

Clearly, as we evaluate at more and more points, we reduce our error, as expected.

Exercise 18.7

We would like to solve

$$\int_0^1 f(x)dx = \int_0^1 p(x) * ln(x)dx.$$

We set up a linearly system of equations for different polynomials for $p(x)$. We would like to determine the best possible weights to solve a polynomial of highest accuracy.

$$p(x) = 1 : \int_0^1 ln(x)dx = w_1 + w_2$$
$$p(x) = x : \int_0^1 xln(x)dx = w_1x_1 + w_2x_2$$

Solve for the left hand sides by using integration by parts:

$$\begin{aligned}
&\int_0^1 ln(x)dx \\
u = ln(x) \quad &v = x \\
du = 1/x \quad &dv = dx \\
&uv - \int_0^1 vdu \\
&= ln(x)x - \int_0^1 dx \\
&= ln(x)x - x|_0^1 \\
= ln(1) * 1 - 1 - ln(0) * 0 - 0 \\
&= -1
\end{aligned}$$

Similarly,

$$\begin{aligned}
&\int_0^1 xln(x)dx \\
u = ln(x) \quad v = 1/2x^2 \\
du = 1/x \quad dv = x \\
&uv - \int_0^1 vdu \\
= 1/2ln(x)x^2 - &\int_0^1 1/2 * x^2 * 1/xdx \\
&= 1/2ln(x)x^2 - 1/4x^2|_0^1 \\
&= -1/4
\end{aligned}$$

With $x_1 = 1/4$ and $x_2 = 3/4$, we solve

$$\begin{aligned}
-1 &= w_1 + w_2 \\
-1/4 &= 1/3 * w_1 + 2/3 * w_2
\end{aligned}$$

We obtain solutions $w_1 = 1/4$ and $w_2 = -5/4$. Then,

$$\begin{aligned}
\int_0^1 f(x)dx &\approx \sum_{j=1}^{N} p(x_j) * w_j \\
&= e^{(1/4)^2} * 1/4 + e^{(3/4)^2} * -5/4 \\
&\approx -1.9105.
\end{aligned}$$

This is pretty decent.