Homework 4 - Solutions
MA/CS 375, Fall 2005

1. In this problem we approximate the derivative of a function defined on a uniform grid on an interval (a, b), i.e. for the grid

$$x_k = a + k\frac{(b-a)}{n}, \quad k = 0, 1, ..., b$$

(a) Show (using Taylor series) that the formula

$$f'(x_k) - \frac{f(x_{k+1}) - f(x_{k-1})}{2h} = \frac{h^2}{6}f'''(\xi_k), \quad \xi_k \in (x_{k-1}, x_{k+1})$$

is valid for all interior grid points, i.e. $k = 1, 2, ..., n-1$.

(b) For the end points, use Taylor series to show that

$$f'(x_0) - \frac{-3f(x_0) + 4f(x_1) - f(x_2)}{2h} = O(h^2)$$

$$f'(x_n) + \frac{-3f(x_n) + 4f(x_{n-1}) - f(x_{n-2})}{2h} = O(h^2)$$

(c) Write a code that uses the above formulas to produce a second order accurate approximation for the derivative of the function

$$f(x) = \cos(10x) - 3\sin(13\pi x)$$

on the interval $-1 \le x \le 1$, with an error no larger than tol $= 10^{-6}$. You must estimate the third derivative of your function using a worst case scenario and choose the stepsize h based on your estimates. You must compute the exact derivative at the same grid points and produce a plot of your absolute error.

Solutions: Let $f_k \equiv f(x_k)$ and $h = x_{k+1} - x_k$. The full Taylor series at points $x_{k\pm1}$ are

$$f_{k\pm1} = \sum_{j=0}^{\infty} \frac{(\pm h)^j}{j!} f_k^{(j)}, \quad f_{k+1} - f_{k-1} = \sum_{j=0}^{\infty} \frac{(\pm h)^j}{j!} f_k^{(j)} = 0 \text{ for } j \text{ even}$$

Since only odd values of j contribute,

$$f_{k+1} - f_{k_1} = \sum_{j=0}^{\infty} \frac{2h^{2j+1}}{(2j+1)!} f_k^{(2j+1)} = 2hf_k' + \frac{1}{3}h^3 f_k''' + \frac{1}{60}h^5 f_k^{(5)} + \cdots$$

$$f_k' - \frac{f_{k+1} - f_{k-1}}{2h} = -\frac{h^2}{6}f_k''' - \frac{h^4}{120}f_k^{(5)} + \cdots$$

Applying the mean value theorem,

$$f'(x_k) - \frac{f(x_{k+1}) - f(x_{k-1})}{2h} = \frac{h^2}{6} f'''(\xi_k), \quad \xi_k \in (x_{k-1}, x_{k+1})$$

The non-symmetric terms are obtains similarly

$$f_1 = f_0 + hf_0' + \frac{h^2}{2} f_0'' + O(h^3), \quad f_2 = f_0 + 2hf_0' + \frac{(2h)^2}{2} f_0'' + O(h^3)$$

$$4f_1 - f_2 = -3f_0 + 2hf_0' + O(h^3) \Rightarrow f_0' - \frac{3f_0 + 4f_1 - f_2}{2h} = O(h^2)$$

For the right end-point

$$f_{n-1} = f_n - hf_n' + \frac{h^2}{2} f_n'' + O(h^3), \quad f_{n-2} = f_n - 2hf_n' + \frac{(-2h)^2}{2} f_n'' + O(h^3)$$

$$4f_{n-1} - f_{n-2} = 3f_n - 2hf_n' + O(h^3) \Rightarrow f_n' + \frac{-3f_n + 4f_{n-1} - f_{n-2}}{2h} = O(h^2)$$

The error bound for the interior points is

$$E_h^{\text{int}} \leq \frac{h^2}{6} \sup_{\xi \in [x_1, x_{n-1}]} |f'''(\xi)|$$

and for the boundary points

$$E_h^{\text{bdry}} \leq \frac{h^2}{3} \max\left(\sup_{\xi \in [x_0, x_2]} |f'''(\xi)|, \sup_{\xi \in [x_{n-2}, x_n]} |f'''(\xi)| \right)$$

The worst case, is at the boundary and since $f'''(\xi)$ does attain its maximum at the boundary, we use this estimate.

$$10^{-6} > 2197\pi^3 h^2, \quad h \leq \sqrt{\frac{10^{-6}}{2197\pi^3}}$$

We can determine the minimum necessary number of grid points:

$$h = \frac{2}{n+1}, \quad n = \frac{2}{h} - 1$$

$$n = \text{ceil}\left(2\sqrt{\frac{2197\pi^3}{10^{-6}}} - 1 \right) = 521999$$

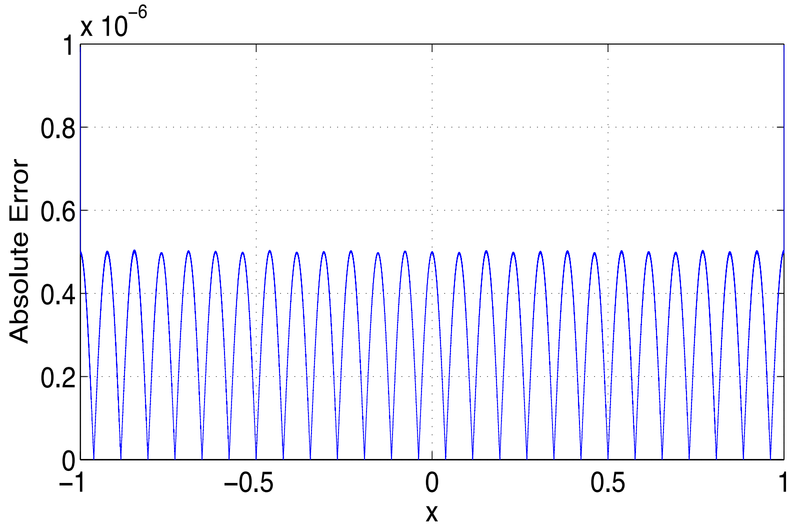
A Matlab script for performing the numerical differentiation is

```
function fp=diff3pt(f,h)
n1=length(f); h2=h*2; n=n1-1; k=2:n; fp=zeros(n1,1);
fp(k)=(f(k+1)-f(k-1))/h2;
fp(1)=[-3*f(1)+4*f(2)-f(3)]/h2; fp(n1)=-[-3*f(n1)+4*f(n)-f(n-1)]/h2;
```

Now we can test to verify that the error estimate is satisfied.

```
>> n=521999; x=linspace(-1,1,n)'; h=x(2)-x(1);
>> f=cos(10*x)-3*sin(13*pi*x);
>> fp=diff3pt(f,h);
>> exact=-10*sin(10*x)-39*pi*cos(13*pi*x);
>> plot(x,abs(exact-fp));
```

The maximum error on the interval occurs at the endpoints and is 9.9763e-07, just under tolerance.



2. Modify one of the programs in the text (midpointc, trapc, or simpsonc) to use the composite Newton's 3/8-rule on a uniform partition to approximate an intergral. On a subinterval $(x_{k-1}, x_k l)$ of width h, the 3/8-rule is defined by:

$$\int_{x_{k-1}}^{x_k} f(x)dx \approx \frac{h}{8}\left(f_{k-1}) + 3f(x_{k-1} + h/3) + 3f(x_{k-1} + 2h/3) + f(x_k)\right)$$

Show that the rule is exact for polynomials of degree 3 or less and use this fact to predict the order of the method. Use the method wih $n = [5\ 10\ 20\ 40\ 60\ 80\ 100]$ subintervals to approximate

$$\int_0^1 \sin(\pi x) e^{2x} dx$$

Using the fact that the exact value of the integral is $\frac{\pi}{4+\pi^2}(e^2+1)$, compute the errors in your approximations and make a loglog plot of the error versus n. Is the result approximately in a straight line? Use polyfit to compute a linear approximation, $\log(\text{error}) = a\log(\text{n}) + b$ to the data. Os the slope, a, what you expect from the error analysis?

Solutions:

A short script implementing the composite Newton's 3/8-rule is

```
function Q=newton38(f,a,b,n)
N=1+3*n; x=linspace(a,b,N)'; h=x(2)-x(1); g=f(x);
Q=(3*h/8)*(g(1)+g(N)+3*sum(g(2:3:N)+g(3:3:N))+2*sum(g(4:3:N-1)));
```

Where `n` is the number of intervals, `a` and `b` are the limits of integration, and `f` is the integrand. Let $p(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. The degree of exactness of the rule can be observed, by computing the integral

$$\int_0^1 p(x)dx = \frac{a_3}{4} + \frac{a_2}{3} + \frac{a_1}{2} + a_0$$

Using the 3/8 rule,

$$Q = \frac{1}{8}\left[p(0) + 3p(1/3) + 3p(2/3) + p(1)\right]$$

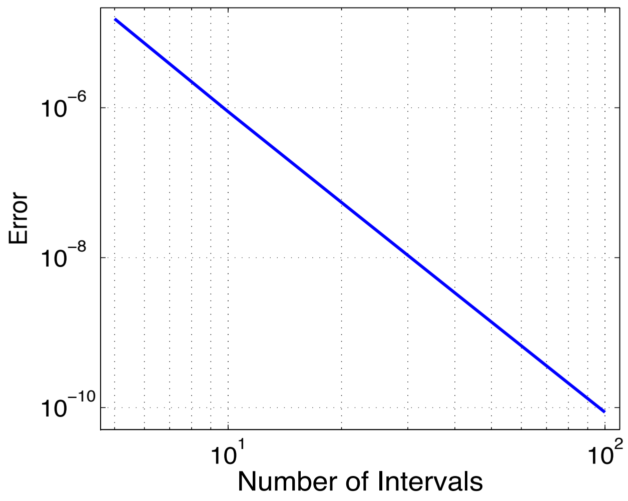
The quadrature sum can be expressed using matrix-vector representation

$$Q = \frac{1}{8}\begin{pmatrix} 1 & 3 & 3 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 & 1 \\ (\frac{1}{3})^3 & (\frac{1}{3})^2 & (\frac{1}{3}) & 1 \\ (\frac{2}{3})^3 & (\frac{2}{3})^2 & (\frac{2}{3}) & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix}$$

$$Q = \begin{pmatrix} \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 1 \end{pmatrix}\begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \frac{a_3}{4} + \frac{a_2}{3} + \frac{a_1}{2} + a_0 = \int_0^1 p(x)dx$$

The 4×4 matrix above, is called a Vandermonde matrix and it relates the integration rule on the grid points to the integration in coefficient representation. Since the 3/8 rule is exact for cubic polynomials, but does not contain a node in the midde of the interval, the error term must be order 4. A script for testing the rule is

```
function err=newton38test(n)
f=@(x) sin(pi*x).*exp(2*x);
Q=newton38(f,0,1,n);
exact=pi*(exp(2)+1)/(4+pi^2);
err=abs(exact-Q);
```



Using `polyval`, the coefficients are $a = -4.0282$ and $b = -4.6331$. The error decreases with order 4 as the grid is refined as one would reasonably expect from the order of the theoretical error term.

3. The four-point Gauss-Lobatto formula for approximating integrals takes the form:

$$\int_{x_{k-1}}^{x_k} f(x)dx \approx h(w_1 f(x_{k-1}) + w_2 f(x_{k-1} + c_1 h) + w_2 f(x_{k-1} + c_2 h) + w_4 f(x_k)), \quad h = x_k - x_{k-1}$$

The weights, w_j, $j = 1, ..., 4$ and the constants c_1, c_2 so that the formula is exact for polynomials of degree 5. Find them. What is the order of the resulting method?

Solution: Using the interval $x \in [-1, 1]$ to simplify the alegebra, we can consider the polynomial function

$$p(x) = \sum_{j=0}^{5} a_j x^j$$

The exact integral of this function is

$$\int_{-1}^{1} \sum_{j=0}^{5} a_j x^j dx = \sum_{j=0}^{5} a_j \int_{-1}^{1} x^j dx == \sum_{j=0}^{5} \frac{a_j}{j+1}[1-(-1)^{j+1}]$$

Since the powers $1, 3, 5$ must all integrate to zero, we can infer there is a symmetry to the quadrature rule. In particular, the two unknown nodes are actually reflections about the origin, and the weights must also be symmetric. Further more, the weights must add up to 2 since this is what the zeroth order term integrates to over $[-1, 1]$. Using these facts, the quadrature rule simplifies to

$$\int_{-1}^{1} p(x)dx = (1-w)p(-1) + wp(-\xi) + wp(\xi) + (1-w)p(1)$$

Now we need two different relationships to uniquely determine w and ξ. Since we have already used information from the constant, linear, cubic, and quintic terms to obtain this expression, the quadratic and quartic are all that remains.

$$\int_{-1}^{1} x^2 dx = 2(1-w) + 2w\xi^2 = \frac{2}{3}, \quad \int_{-1}^{1} x^4 dx = 2(1-w) + 2w\xi^4 = \frac{2}{5}$$

Rearranging, gives

$$w = -\frac{2}{3(\xi^2-1)} = -\frac{4}{5(\xi^4-1)}$$

One obtains an expression for the unknown node, ξ

$$5(\xi^2)^2 - 6\xi^2 + 1 = 0, \quad \xi^2 = 1, \ \frac{1}{5}$$

From this the weight is determined

$$w = \frac{2}{3(\frac{1}{5}-1)} = \frac{5}{6}$$

Therefore, the four-point Gauss-Lobatto quadrature rule for $[-1, 1]$ is

$$\int_{-1}^{1} f(x)dx \approx \frac{1}{6}f(-1) + \frac{5}{6}f(-\frac{1}{\sqrt{5}}) + \frac{5}{6}f(\frac{1}{\sqrt{5}}) + \frac{1}{6}f(1)$$

We can now scale this to the arbitrary interval $[x_{k-1}, x_k]$ using a linear mapping.

$$w_1 = w_4 = \frac{h}{12}, \quad w_2 = w_3 = \frac{5h}{12}$$

$$c_1 = \frac{1}{2} - \frac{\sqrt{5}}{10}, \quad c_2 = \frac{1}{2} + \frac{\sqrt{5}}{10}$$

By construction, the method is exact for quintic polynomials. To leading order, the error will scale with the sixth derivative of $f(x)$. (*Note:* For higher order quadrature, the nodes and weights are typically computed by using properties of the Legendre polynomials as they are the polynomial basis orthogonal with respect to the constant weighting function.)

4. The adaptive quadrature code given in the text, estimates ther error in Simpson's rule by performing two integrations over the interval A. The first uses the simple Simpson's rule, with the nodes $[a, (a+b)/2, b]$, i.e. with stepsize $H_1 = b - a$ to compute the approximation I_s. The second uses the composite Simpson's rule with two subintervals, i.e. nodes $[a, (3a+b)/4, (a+b)/2, (a+3b)/4, b]$ ans stepsize $H_2 = (b-a)/2$, to compute the (more accurate) approximation I_s^c.

   The text uses the two approximations to estimate the error and, if it turns out that to be less than a predefined tolerance, it uses the more accurate resut of the two, i. e. I_s^c, as the actual estimate for the integral in the interval A. This problem is about improving this calculation through the judicious uses of both results, I_s and I_s^c.

   It can be shown that the error in Simpson's method for integrating a function that is differentiable at least five times can be actually written as

$$I[f] - I_s^c[f; M] = c_1 \left(\frac{b-a}{M}\right)^4 + c_2 \left(\frac{b-a}{M}\right)^5 + \cdots$$

   where c_1, c_2 are constants, independent of M (for functions that have more derivatives in the interval of integration, we can add more terms to that expression, but that is not important here). The leading term in this expression is of order 4, which is the order of the method. In this problem, we learn how to get a more accurate result using the work already performed. To that purpose:

   (a) Find a linear combination of the two results that has leading error term of order 5; that is you must find constants A_1 and A_2 such that

$$I[f] - (A_1 I_s^c + A_2 I_s) = c_2' \left(\frac{b-a}{M}\right)^5 + \cdots$$

   The idea is to combine the two results in such a way so that the leading error term is cancelled.

   (b) Now alter the program `simpadpt` in the text, p. 98, to use this improved estimate of the integral over A instead of the (less accurate) estimate I_s^c. Demonstrate your code by producing a more accurate computation of the integral in **example 4.3**. Your computation should employ the same tolerance ($tol = 10^{-5}$ and minimum stepsize ($hmin = 10^{-3}$) as the values in the text and therefore end up using the same number of intervals. Nevertheless, your result ought to be closer to the exact value

$$\int_{-1}^{1} e^{-10(x-1)^2} dx = 0.28024956081990$$

than the one found by the program `simpadpt`.

Solutions: The coefficients must satisfy the linear equations

$$\begin{pmatrix} \frac{1}{16} & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad A_1 = \frac{16}{15}, \quad A_2 = -\frac{1}{15}$$

The first row equation cancels the fourth order term in the error and the second row equation requires that the integration of a constant have the same value. Considering a 5-point interval, the new weights for integration are obtained by combining the two rules.

$$I_{\text{new}} = \frac{h}{45}\left[7f(0) + 32f(h) + 12f(2h) + 32f(3h) + 7f(4h)\right]$$

Only one line of `simpadpt` needs to be modified to increase the order. Modifying the line

```
ISc=0.5*L*sum(fx.*[w;w]);
```

to

```
ISc=(16*(0.5*L*sum(fx.*[w;w]))-IS)/15;
```

is all that is needed. This change reduces the error from 3.09320059183049e-06 to 1.43952124964652e-06, or roughly halves the error from the original method.