Problem 1 In class we proved the Euler-Maclaurin summation formula

$$\int_0^1 f(x)\mathrm{d}x = \frac{1}{2}\left(f(0)+f(1)\right) + \sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(1) - f^{(2m-1)}(0)\right)$$

for some unknown constants b_m independent of f.

(a) Find a recursive formula for b_m by evaluating both sides for $f(x) = e^{\lambda x}$ where λ is a parameter.

(b) Compute $b_1, b_2, b_3, \ldots, b_{10}$.

(c) Compound the formula to show

$$\int_0^n f(x)\mathrm{d}x = \frac{1}{2}f(0)+f(1)+f(2)+\cdots+f(n-1)+\frac{1}{2}f(n)+\sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(n) - f^{(2m-1)}(0)\right).$$

(d) Use the Euler-Maclaurin formula to show that

$$\sum_{j=1}^{n} j^k = P_{k+1}(n)$$

is a degree-$(k+1)$ polynomial in n. Example:

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}.$$

(e) Show that the error in the trapezoidal rule satisfies

$$\int_0^1 f(x)\mathrm{d}x - h\left(\frac{1}{2}f(0) + f(h) + f(2h) + \cdots + f((n-1)h) + \frac{1}{2}f(nh)\right)$$

$$= \sum_{m=1}^{\infty} b_m h^{2m} \left(f^{(2m-1)}(1) - f^{(2m-1)}(0)\right).$$

Solution 1 (15 pts)

1. Plugging in $f(x) = e^{\lambda x}$ gives the formula

$$\frac{e^{\lambda}-1}{\lambda} = \frac{1}{2}\left(e^{\lambda}+1\right) + \sum_{m=1}^{\infty} b_m \lambda^{2m-1}\left(e^{\lambda}-1\right).$$

Multiply both sides by $\lambda(e^{\lambda}-1)^{-1}$ to yield

$$\frac{-\lambda}{e^{\lambda}-1} = -1 + \frac{\lambda}{2} + \sum_{m=1}^{\infty} b_m \lambda^{2m}. \tag{1}$$

Therefore b_m is the Taylor coefficient of the function

$$f(\lambda) = \frac{-\lambda}{e^\lambda - 1}$$

corresponding to λ^{2m}. Let's set this up in the form

$$(e^\lambda - 1)\left[-1 + \frac{\lambda}{2} + \sum_{m=1}^{\infty} b_m \lambda^{2m}\right] = -\lambda$$

and define a new power series $\sum_j \alpha_j \lambda^j$ where $\alpha_0 = -1$, $\alpha_1 = 1/2$, and $b_m = \alpha_{2m}$. Then divide the power series for $e^\lambda - 1$ by λ to get the relation

$$\left(\sum_{k=0}^{\infty} \frac{\lambda^k}{(k+1)!}\right)\left(\sum_{j=0}^{\infty} \alpha_j \lambda^j\right) = -1.$$

We may rearrange this sum in the form

$$\sum_{s=0}^{\infty}\sum_{t=0}^{s} \alpha_t \lambda^s \frac{1}{(s-t+1)!} = -1,$$

and equating terms gives

$$\sum_{t=0}^{s} \frac{\alpha_t}{(s-t+1)!} = \begin{cases} -1 & s = 0 \\ 0 & \text{otherwise} \end{cases}$$

Equation (1) implies that $\alpha_j = 0$ for all odd j greater than 1, so we can eliminate those indices, separate out α_0 and α_1 from the rest of the sum, and then substitute $\alpha_{2j} = b_j$ to get

$$\frac{-1}{(2m+1)!} + \frac{1}{2(2m)!} + \sum_{k=1}^{m} \frac{b_k}{(2(m-k)+1)!} = 0.$$

Subtracting the final term of the sum from both sides gives the desired recurrence:

$$b_m = \frac{1}{(2m+1)!} - \frac{1}{2(2m)!} - \sum_{k=1}^{m-1} \frac{b_k}{(2(m-k)+1)!} \tag{2}$$

2. If we program relation (2) in Matlab then we will start losing precision after the first few numbers and won't get exact rational numbers. In any case, the exact values of the first ten numbers $b_1, \ldots, b_{10}$ are

$$b_1 = -\frac{1}{12}, b_2 = \frac{1}{720}, b_3 = -\frac{1}{30240},$$
$$b_4 = \frac{1}{1209600}, b_5 = -\frac{1}{47900160}, b_6 = \frac{691}{1307674368000},$$
$$b_7 = -\frac{1}{74724249600}, b_8 = \frac{3617}{10670622842880000}, b_9 = -\frac{43867}{5109094217170944000},$$
$$b_{10} = \frac{174611}{802857662698291200000}.$$

3. Rearranging terms in the Euler-Maclaurin summation formula gives

$$\sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(1) - f^{(2m-1)}(0)\right) = \int_0^1 f(x)\ dx - \frac{1}{2}\left(f(0) + f(1)\right)$$

Applying this formula to the function $f(x-k)$ for some integer k shifts the formula over

$$\sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(k) - f^{(2m-1)}(k-1)\right) = \int_{k-1}^{k} f(x)\,dx - \frac{1}{2}\left(f(k-1) + f(k)\right)$$

Adding n copies of this equation, with $k = 1, ..., n$, gives the following:

$$\sum_{k=1}^{n}\sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(k) - f^{(2m-1)}(k-1)\right) = \sum_{k=1}^{n}\int_{k-1}^{k} f(x)\,dx - \frac{1}{2}\left(f(k-1) + f(k)\right)$$

Simplifying both sides, and seeing that the left side telescopes, gives

$$\sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(n) - f^{(2m-1)}(0)\right) = \int_0^n f(x)\,dx - \frac{1}{2}f(0) - f(1) - ... - f(n-1) - \frac{1}{2}f(n)$$

Bringing the function evaluations to the left side of the equation yields

$$\frac{1}{2}f(0) + f(1) + ... + f(n-1) + \frac{1}{2}f(n) \sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(n) - f^{(2m-1)}(k)\right) = \int_0^n f(x)\,dx$$

which is exactly what is to be proved.

4. Let $f(x) = x^k$. Then the derivatives $f^{(2m-1)}(x)$ of f are all zero for $2m - 1 > k$ or $m > \lfloor (k+1)/2 \rfloor$, so the above formula becomes

$$\int_0^n x^k\,dx = \frac{1}{2}0^k + 1^k + 2^k + \ldots + (n-1)^k + \frac{1}{2}n^k + \sum_{m=1}^{\lfloor (k+1)/2 \rfloor} b_m \left(f^{(2m-1)}(n) - f^{(2m-1)}(0)\right)$$

where 0^k is 0 unless $k = 0$ and 1 if $k = 0$. Evaluating the integral and taking the derivatives gives the formula

$$\frac{n^{k+1}}{k+1} = \frac{1}{2}0^k + 1^k + 2^k + \ldots + (n-1)^k + \frac{1}{2}n^k + \sum_{m=1}^{\lfloor (k+1)/2 \rfloor} b_m \left(\frac{k!}{(k-2m+1)!}\left(n^{k-2m+1} - 0^{k-2m+1}\right)\right)$$

Adding $\frac{1}{2}(0^k + n^k)$ to both sides and bringing the sum to the left side gives

$$\frac{n^{k+1}}{k+1} + \frac{1}{2}n^k - \sum_{m=1}^{\lfloor (k+1)/2 \rfloor} b_m \left(\frac{k!}{(k-2m+1)!}\left(n^{k-2m+1} - 0^{k-2m+1}\right)\right) = \sum_{j=0}^{n} j^k$$

Clearly the left side of this equation is a degree $k+1$ polynomial in n.

5. Given a function f, define $g(x) := f(hx)$ where $h = \frac{1}{n}$. Then the Euler-Maclaurin formula applied to g is

$$\int_0^n g(x)\,dx = \frac{1}{2}f(0h)+f(1h)+f(2h)+\ldots+\frac{1}{2}f(nh)+\sum_{m=1}^{\infty} b_m \left(g^{(2m-1)}(n) - g^{(2m-1)}(0)\right)$$

By the chain rule, $g^{(2m-1)}(x) = h^{2m-1}f^{(2m-1)}(hx)$, so scaling the variable of integration by h gives

$$\frac{1}{h}\int_0^1 f(x)\,dx = \frac{1}{2}f(0h)+f(1h)+f(2h)+\ldots+\frac{1}{2}f(nh)+\sum_{m=1}^{\infty} b_m \frac{h^{2m}}{h}\left(f^{(2m-1)}(hn) - f^{(2m-1)}(0h)\right)$$

Multiplying both sides by h, observing that $hn = 1$, and bringing the evaluations of f to the left side gives

$$\int_0^1 f(x)\,dx - h\left(\frac{1}{2}f(0h) + f(1h) + \ldots + f\left(h(n-1)\right) + \frac{1}{2}f(1)\right) = \sum_{m=1}^{\infty} b_m h^{2m}\left(f^{(2m-1)}(1) - f^{(2m-1)}(0)\right)$$

This is exactly what was to be shown.

Problem 2 Write a matlab program `ectr.m` of the form

```
function w = ectr(n, k)
% n : quadrature points are 0 ... n
% k: degree of precision
```

which produces the weight vector $(w_0, \ldots, w_n)$ containing endpoint-corrected trapezoidal weights of even order $k = 2, 4, \ldots, 10$: for given $n \geq 2k$. Your code should combine previous codes for the differentiation matrix, the Bernoulli numbers b_1 through b_{10}, and the Euler-Maclaurin summation formula. Use it to complete the following table of endpoint-corrected trapezoidal weights of even order $k = 2, 4, \ldots, 10$:

k	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	$\cdots$
2	1/2	1	1	1	1	1	1	1	$\cdots$
4	9/24	28/24	23/24	1	1	1	1	1	$\cdots$
6						1	1	1	$\cdots$
$\vdots$									

Check the order of accuracy of the weights on

$$\int_0^1 e^x dx$$

for $h = 1/32, \ldots, 1/1024$ and even $k = 2, \ldots, 10$.

Solution 2 (15 pts)

The trapezoidal rule is accurate of order $k = 2$. A higher-order endpoint-corrected rule can be built by approximating the first few terms from the summation

$$\sum_{m=1}^{\infty} b_m \left(f^{(2m-1)}(n) - f^{(2m-1)}(0) \right)$$

with sufficient accuracy. The derivatives evaluated at 0 can be approximated to order $k - 2m$ accuracy by the formula

$$f^{(2m-1)}(0) \approx p^{(2m-1)}(0) = \sum_{j=0}^{n} \delta_{nj}^{2m-1}(0) f_j$$

taken from problem set 3, where $f_j = f(x_j) = f(j)$ and $n = k - 1$. If the points x_j used in evaluating derivatives at zero do not overlap with the x_j used to evaluate derivatives at n, the resulting integration rule has weights

$$w_j = 1 - \sum_{m=\lceil \frac{j}{2} \rceil + 1}^{\frac{k}{2}-1} b_m \delta_{2m+1,j}^{2m-1}(0)$$

where $\delta_{10}^{-1}(0) = 1$ and $b_0 = \frac{1}{2}$ are set for convenience. The δ values are determined using the recurrence relation in the embedded code. These coefficients should be exactly

k	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
2	$\frac{1}{2}$	1	1	1	1	1	1	1	1	1
4	$\frac{9}{24}$	$\frac{28}{24}$	$\frac{23}{24}$	1	1	1	1	1	1	1
6	$\frac{545}{1440}$	$\frac{1662}{1440}$	$\frac{1404}{1440}$	$\frac{1426}{1440}$	$\frac{1443}{1440}$	1	1	1	1	1
8	$\frac{2281}{6048}$	$\frac{7024}{6048}$	$\frac{5797}{6048}$	$\frac{6112}{6048}$	$\frac{5975}{6048}$	$\frac{6080}{6048}$	$\frac{6043}{6048}$	1	1	1
10	$\frac{912809}{2419200}$	$\frac{2808090}{2419200}$	$\frac{2320958}{2419200}$	$\frac{2443218}{2419200}$	$\frac{2390800}{2419200}$	$\frac{2431566}{2419200}$	$\frac{2417410}{2419200}$	$\frac{2419143}{2419200}$	$\frac{2419206}{2419200}$	1

See embedded MATLAB file for floating-point computation of above weights. Note that the weight vector for a given k should not exactly integrate a polynomial of degree $k-1$. While the trapezoidal rule ($k = 2$) integrates linear polynomials exactly but not quadratics, the 4th-order rule is based on quadratic interpolation at the end intervals so will not be quite exact on cubics. This does not prevent 4th-order accuracy because degree of precision is sufficient but not necessary for the order of accuracy of a compound rule.

We verify our rules on the following polynomials, each of which integrates to zero over the interval [0,100]:

$$f_2(x) = \frac{-3x^2 + 300x - 5000}{1000}$$

$$f_4(x) = \frac{3x^4 - 600x^3 + 34800x^2 - 480000x - 2000000}{400000}$$

$$f_6(x) = -210\left(\frac{x}{100}\right)^6 + 210\left(\frac{x}{100}\right)^5 - 5$$

$$f_8(x) = -54\left(\frac{x}{100}\right)^8 + 66\left(\frac{x}{100}\right)^5 - 5$$

$$f_{10}(x) = 11\left(\frac{x}{100}\right)^{10} - 30\left(\frac{x}{100}\right)^9 + 28\left(\frac{x}{100}\right)^3 - 5$$

Using the implementation embedded, numerical integration of these polynomials gives the following results for $h = 1$:

k	2	4	6	8	10
f_2	5.0000×10^{-02}	2.6645×10^{-15}	1.7764×10^{-15}	2.6645×10^{-15}	5.3291×10^{-15}
f_4	1.9998×10^{-01}	4.6750×10^{-04}	1.3767×10^{-14}	1.3323×10^{-14}	7.1054×10^{-15}
f_6	1.7498×10^{-01}	3.2215×10^{-04}	2.0840×10^{-07}	4.4409×10^{-16}	3.3751×10^{-14}
f_8	8.4980×10^{-02}	3.5740×10^{-04}	4.9183×10^{-07}	1.9290×10^{-10}	5.3291×10^{-15}
f_{10}	6.3323×10^{-02}	1.8277×10^{-04}	1.6866×10^{-07}	9.2007×10^{-11}	2.6068×10^{-13}

From these results, it is clear that the order-k numerical integration scheme integrates polynomials of degree $k-2$ exactly modulo floating point errors. This

agrees with our derivation of the 4th-order rule in class by quadratic interpolation. Running the embedded code shows that when the rule is not exact, the ratio of successive errors as h is halved is very close to 4, 16 and 64 for $k = 2$, 4 and 6. This suggests that we have the correct order of accuracy. For $k = 8$ and 10 the errors are so small that we cannot measure the order of convergence on these integrands.

Problem 3 (a) Find an exact formula for the quintic polynomial $P_5(x) = x^5 + \cdots$ such that

$$\int_{-1}^{1} P_5(x)q(x)dx = 0$$

for any quartic polynomial q.

(b) Find exact formulas for the five roots x_1, x_2, x_3, x_4, x_5 of the equation $P_5(x) = 0$.

(c) Find exact formulas for the integration weights w_1, w_2, w_3, w_4, w_5 such that

$$\int_{-1}^{1} q(x)dx = \sum_{j=1}^{5} w_j q(x_j)$$

exactly whenever q is a polynomial of degree 5.

(d) Given any real numbers $a < b$, find exact formulas for points $y_j \in [a, b]$ and weights $u_j > 0$ such that

$$\int_{a}^{b} q(x)dx = \sum_{j=1}^{5} u_j q(y_j)$$

whenever q is a polynomial of degree 5.

(e) Explain why each of the three factors in the error estimate

$$\int_a^b f(x)dx - \sum_{j=1}^{5} u_j f(y_j) = C_{10} f^{(10)}(\xi) \int_a^b (y-y_1)^2(y-y_2)^2(y-y_3)^2(y-y_4)^2(y-y_5)^2 dy$$

is inevitable and determine the exact value of the constant C_{10}.

Solution 3 (15 pts)

(a)

In class we constructed polynomials $P_0(x) = 1$, $P_1(x) = x$, and

$$P_2(x) = xP_1(x) - \frac{1}{3}P_0(x) = x^2 - \frac{1}{3}$$

such that $\int P_j q\, dx = 0$ whenever q had lower degree than P_j. Since each P_j begins with x^j, the P_j's form a basis for quadratic polynomials. Hence we seek $P_3(x) = xP_2(x) - cP_1(x)$ where c is chosen to make

$$0 = \int_{-1}^{1} P_3(x) \cdot P_0(x)\, dx = \int_{-1}^{1} P_3(x) \cdot P_1(x)\, dx = \int_{-1}^{1} P_3(x) \cdot P_2(x)\, dx.$$

Here P_0 and P_2 are automatic by parity since P_3 is odd and they are even. So c must satisfy

$$\int_{-1}^{1} (xP_2(x) - cP_1(x))P_1(x)\, dx = 0$$

or

$$c = \frac{\int_{-1}^{1} xP_2(x)P_1(x)\,dx}{\int_{-1}^{1} P_1(x)^2\,dx} = \frac{\int_{-1}^{1} x^4 - \frac{1}{3}x^2\,dx}{\int_{-1}^{1} x^2\,dx} = \frac{\frac{8}{45}}{\frac{2}{3}} = \frac{4}{15}.$$

Thus $P_3(x) = x^3 - \frac{3}{5}x = \frac{5x^3-3x}{5}$.

Proceeding similarly, we find

$$P_5(x) = x^5 - \frac{10}{9}x^3 + \frac{5}{21}x.$$

(b)

We can factor $P_5(x)$ so that the roots are $t_0 = 0$ and the positive and negative square roots t_1, t_2 and t_3, t_4 of

$$y_1 = \frac{5}{9} + \frac{2\sqrt{10}}{9\sqrt{7}}$$

and

$$y_2 = \frac{5}{9} - \frac{2\sqrt{10}}{9\sqrt{7}}$$

respectively.

(c)

Since the rule is accurate for degree-9 polynomials we must have *a fortiori*

$$\int_{-1}^{1} L_j(x)\,dx = \sum_{k=1}^{5} w_k L_j(x_k) = w_j$$

for each quartic Lagrange basis polynomial

$$L_j(x) = \frac{\prod_{k\neq j}(x - x_k)}{\prod_{k\neq j}(x_j - x_k)}.$$

It is also true that

$$\int_{-1}^{1} L_j^2(x)\,dx = \sum_{k=1}^{5} w_k L_j^2(x_k) = w_j$$

which is convenient for numerical integration because all values are positive, but less convenient for hand computation since the degree doubles. Tedious computation gives the weights

$$w_0 = \frac{128}{225},$$

$$w_1, w_2 = \pm\sqrt{\frac{322 - 13\sqrt{70}}{900}},$$

$$w_3, w_4 = \pm\sqrt{\frac{322 + 13\sqrt{70}}{900}}.$$

(d)

The change of variables

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

sends the interval $[-1, 1]$ to $[a, b]$. The change of variable $dy = \frac{b-a}{2}dx$ tells us that

$$\int_a^b f(y)\,dy = \frac{b-a}{2}\int_{-1}^1 g(x)\,dx$$

where $g(x) = f\left(\frac{a+b}{2} + x\frac{b-a}{2}\right)$ is a polynomial of the same degree as f.

Since the transformation $x \longmapsto y$ preserves polynomial degree we can re-use our weights from the previous parts.

$$\int_a^b f(y)\,dy = \frac{b-a}{2}\int_{-1}^1 g(x)\,dx = \frac{b-a}{2}\left[w_1 g(x_1) + w_2 g(x_2) + w_3 g(x_3) + \cdots\right]$$

This means we can use the weights

$$\boxed{u_j = \frac{b-a}{2}w_j}$$

(we have $u_j > 0$ since $b > a, w_j > 0$) and the values $g(x_j) = f\left(\frac{a+b}{2} + x_j\frac{b-a}{2}\right)$ mean we can use the nodes

$$\boxed{y_j = \frac{a+b}{2} + x_j\frac{b-a}{2}}.$$

(e) Gaussian quadrature with five nodes can be seen as arising from a Hermite interpolation, but a very special one. If we interpolate

$$H(x) = H_1(x)f(x_1) + H_2(x)f(x_2) + H_3(x)f(x_3) + \cdots + \widehat{H_1}(x)f'(x_1) + \widehat{H_2}(x)f'(x_2) + \widehat{H_3}(x)f'(x_3) + \cdots$$

and then approximate

$$\begin{aligned}\int_{-1}^1 f(x)\,dx &\approx \int_{-1}^1 H(x)\,dx \\ &= f(x_1)I_1 + f(x_2)I_2\,dx + f(x_3)I_3 + \cdots + f'(x_1)\widehat{I_1} + f'(x_2)\widehat{I_2} + f'(x_3)\widehat{I_3} + \cdots\end{aligned}$$

where

$$I_j = \int_{-1}^1 H_j(x)\,dx, \quad \widehat{I_j} = \int_{-1}^1 \widehat{H_j}(x)\,dx.$$

Gaussian quadrature corresponds to the unique choice $x_1 < x_2 < x_3 < x_4 < x_5$ such that

$$\widehat{I_1} = \widehat{I_2} = \widehat{I_3} = \cdots = 0.$$

This allows $\int_{-1}^{1} f(x)\,dx$ to be approximated without the ability to compute $f'(x_0)$ for any particular value.

From our approximation, we have

$$f(x) - H(x) = \frac{f^{(10)}(\xi(x))}{10!}(x - x_1)^2(x - x_2)^2(x - x_3)^2 \cdots$$

for $\xi(x) \in (-1, 1)$. The I_j and $\widehat{I_j}$ values mentioned above mean that integrating the error term above gives

$$\int_{-1}^{1} \frac{f^{(10)}(\xi(x))}{10!}(x - x_1)^2(x - x_2)^2(x - x_3)^2 \cdots dx.$$

Since $(x - x_1)^2(x - x_2)^2(x - x_3)^2 \cdots$ is non-negative, the mean value theorem for integrals can be applied and we find

$$\int_{-1}^{1} \frac{f^{(10)}(\xi(x))}{10!}(x-x_1)^2(x-x_2)^2(x-x_3)^2 \cdots dx = \frac{f^{(6)}(\xi)}{6!} \int_{-1}^{1} (x-x_1)^2(x-x_2)^2(x-x_3)^2 \cdots dx$$

for a fixed value $\xi \in (-1, 1)$.

Why are each of the factors in the error estimate inevitable?

- The integral is inevitable because Gaussian quadrature is equivalent to integrating a Hermite interpolating polynomial. Also, the integrand is a polynomial of degree 10 for which the Gaussian integration gives 0 and the integral is positive. Hence testing Gaussian integration on this integrand shows that the error must be proportional to this integral.
- The tenth derivative is inevitable because the error is zero whenever the integrand f is a polynomial of degree 9 or less.
- The coefficient C_{10} is inevitable because the sixth derivative of the integrand in the integral factor introduces a factor of 10!.

Problem 4 Write, test and debug an adaptive 5-point Gaussian integration code `gadap.m` of the form

```
function [int, abt] = gadap(a, b, f, r, tol)
% a,b: interval endpoints with a < b
% f: function handle f(x, r) to integrate
% r: parameters for f
% tol: User-provided tolerance for integral accuracy
% int: Approximation to the integral
% abt: Endpoints and approximations
```

Build a list `abt` $= \{[a_1, b_1, t_1], \ldots, [a_n, b_n, t_n]\}$ of n intervals $[a_j, b_j]$ and approximate integrals $t_j \approx \int_{a_j}^{b_j} f(x, r)dx$, computed with 5-point Gaussian integration. Initialize with $n = 1$ and $[a_1, b_1] = [a, b]$. At each step $j = 1, 2, \ldots$, subdivide interval j into left and right half-intervals l and r, and approximate the integrals t_l and t_r over each half-interval by 5-point Gaussian quadrature. If

$$|t_j - (t_l + t_r)| > \texttt{tol} \max(|t_j|, |t_l| + |t_r|)$$

add the half-intervals l and r and approximations t_l and t_r to the list. Otherwise, increment `int` by t_j. Guard against infinite loops and floating-point issues as you see fit and briefly justify your design decisions in comments.

Solution 4 (15 pts)

```
function [int, abt] = gadap(a, b, f, p, tol)
% a,b: interval endpoints with a < b
% f: function handle f(x, p) to integrate (p for user parameters)
% tol: User-provided tolerance for integral accuracy
% int: Approximation to the integral
% abt: Endpoints and approximations

q = 3;
w = [ 5, 8, 5 ] / 9;
x = [ -sqrt(3/5), 0, sqrt(3/5) ];

% integrate over [a,b]
m = 0.5 * ( a + b );
l = 0.5 * ( b - a );
int = 0.0;
for i = 1 : q
int = int +  w( i ) * f( m + l * x( i ), p );
end
int = l * int;

n = 1;
abt( 1, 1 ) = a;
abt( 2, 1 ) = b;
```

```
abt( 3, 1 ) = int;

j = 0;
int = 0;
while(  j < n )
j = j + 1;

% subdivide [a,b] -> [al,bl] U [ar,br]
al = abt( 1, j );
br = abt( 2, j );
intab = abt( 3, j );

bl = 0.5 * ( al + br );
ar = bl;

%integrate over l
m = 0.5 * ( al + bl );
l = 0.5 * ( bl - al );
intl = 0.0;
for i = 1 : q
intl = intl +  w( i ) * f( m + l * x( i ), p );
end
intl = l * intl;

%integrate over r
m = 0.5 * ( ar + br );
l = 0.5 * ( br - ar );
intr = 0.0;
for i = 1 : q
intr = intr +  w( i ) * f( m + l * x( i ), p );
end
intr = l * intr;

%compare and decide
if( abs( intab - ( intl + intr ) ) > tol * abs( intab ) )
n = n + 1;
abt( 1, n ) = al;
abt( 2, n ) = bl;
abt( 3, n ) = intl;
n = n + 1;
abt( 1, n ) = ar;
abt( 2, n ) = br;
abt( 3, n ) = intr;
else
%j;
int = int + intab;
```

```
end
end

end
```

Problem 5 (a) Show that

$$\int_0^1 x^{-x}\mathrm{d}x = \sum_{n=1}^{\infty} n^{-n}$$

(b) Use the sum in (a) to evaluate the integral in (a) to 12-digit accuracy.

(c) Evaluate the integral in (a) by `ectr.m` to 1, 2, and 3-digit accuracy. Estimate how many function evaluations will be required to achieve p-digit accuracy for $1 \le p \le 12$. Explain the agreement or disagreement of your results with theory.

(d) Approximate the integral $\int_0^1 x^{-x}dx$ using your code `gadap.m`. Tabulate the total number of function evaluations required to obtain p-digit accuracy for $1 \le p \le 10$. Compare your results with the results and estimates for endpoint-corrected trapezoidal integration obtained in (c).

Solution 5 (15 pts)

(a)

First use $x^{-x} = e^{-x\ln x}$ and expand the exponential function in Taylor series, we get

$$\int_0^1 x^{-x}\,dx = \int_0^1 e^{-x\ln x}\,dx = \int_0^1 \sum_{k=0}^{\infty} \frac{1}{k!}(-x)^k(\ln x)^k\,dx = \sum_{k=0}^{\infty}\frac{1}{k!}\int_0^1 (-x)^k(\ln x)^k\,dx,$$

where the change in order of summation and integration is because $x\ln x$ is bounded on $[0,1]$, hence the summation converges uniformly on $[0,1]$. Make the substitution $u = -\ln x$, then $x = e^{-u}$, $dx = -xdu$, and the integration becomes

$$\begin{aligned}&\sum_{k=0}^{\infty}\frac{1}{k!}\int_0^1 (-x)^k(\ln x)^k\,dx = \sum_{k=0}^{\infty}\frac{1}{k!}\int_{\infty}^0 (-x)^{k+1}(-u)^k\,du\\ =&\sum_{k=0}^{\infty}\frac{1}{k!}\int_0^{\infty} e^{-u(k+1)}u^k\,du = \sum_{k=0}^{\infty}\frac{1}{k!(k+1)^k}\int_0^{\infty} e^{-u(k+1)}[(k+1)u]^k\,du.\end{aligned}$$

Make the substitution $t = (k+1)u$ the integration becomes

$$\sum_{k=0}^{\infty}\frac{(-1)^k}{k!}\frac{1}{(k+1)^k}\int_0^{\infty} e^{-t}t^k\,dt = \sum_{k=0}^{\infty}\frac{1}{(k+1)!(k+1)^k}\Gamma(k+1).$$

Through integration by parts, we can see $\Gamma(k+1) = k\Gamma(k)$. With the knowledge that $\Gamma(1) = \int_0^{\infty} e^{-t}\,dt = 1$, we can see $\Gamma(k+1) = k!$. Indeed, Γ is the usual Gamma function. The final expression is

$$\sum_{k=0}^{\infty}\frac{1}{(k+1)^{k+1}} = \sum_{n=1}^{\infty}\frac{1}{n^n}.$$

(b) Write

$$\int_0^1 x^{-x}\,dx = S_N + R_N,$$

where for any integer $N \geq 1$,

$$S_N = \sum_{n=1}^{N} n^{-n}, \quad R_N = \sum_{n=N+1}^{\infty} n^{-n}.$$

We need to choose $N \geq 1$ such that $R_N < 10^{-12}$, which will guarantee that the finite sum S_N approximates the integral up to twelve digits of accuracy.

Consider the ratio of successive terms.

$$\frac{(n+1)^{-(n+1)}}{n^{-n}} = n^{-1}\frac{(n+1)^{-(n+1)}}{n^{-(n+1)}} = n^{-1}\left(1+\frac{1}{n}\right)^{-(n+1)}$$

For $n \geq 1$ the second term $(1+1/n)^{-(n+1)}$ is (crudely) bounded by one for example. Therefore

$$\frac{(n+1)^{-(n+1)}}{n^{-n}} \leq n^{-1}.$$

Furthermore n^{-1} is a decreasing sequence, so if $n \geq N+1$, then the ratio between successive terms is bounded by $(N+2)^{-1}$. Therefore

$$\sum_{n=N+1}^{\infty} n^{-n} \leq (N+1)^{-(N+1)} \sum_{j=0}^{\infty} (N+2)^{-j}.$$

The geometric sum on the right hand side converges for $N \geq 0$,

$$(N+1)^{-(N+1)} \sum_{j=0}^{\infty} (N+2)^{-j} = \frac{N+2}{N+1}(N+1)^{-(N+1)}$$

This shows that

$$R_N \leq (N+2)(N+1)^{-(N+2)},$$

and hence $R_N < 10^{-12}$ if $N \geq 11$. Calculating S_{11} gives the approximation

$$\int_0^1 x^{-x}\,dx \approx 1.291285997062548$$

(c)

The error formula does not solve our problem since the derivatives of x^{-x} are unbounded near zero. We use for example $k = 4$ in `ectr.m`, and obtain the following relation of p-digit accuracy and number of small intervals, N. The code we used to generate this table is `q4c.m`.

p	N
1	2
2	3
3	10
4	32
5	103
6	327
7	1036
8	3279
9	10372

From the table we can see that to improve accuracy by one digit, we need roughly 3 times more N. However, if we wrongly assumed that the 4-th order derivative is bounded, then the error formula tells us that 3 times more N reduces the error by $(\frac{1}{3})^4 \approx 0.01$, which means that the accuracy should been improved by two digits, which is wrong.

Problem 6 Implement, debug and test a MATLAB function `pleg.m` of the form

```
function p = pleg(t, n)
% t: evaluation point
% n: degree of polynomial
```

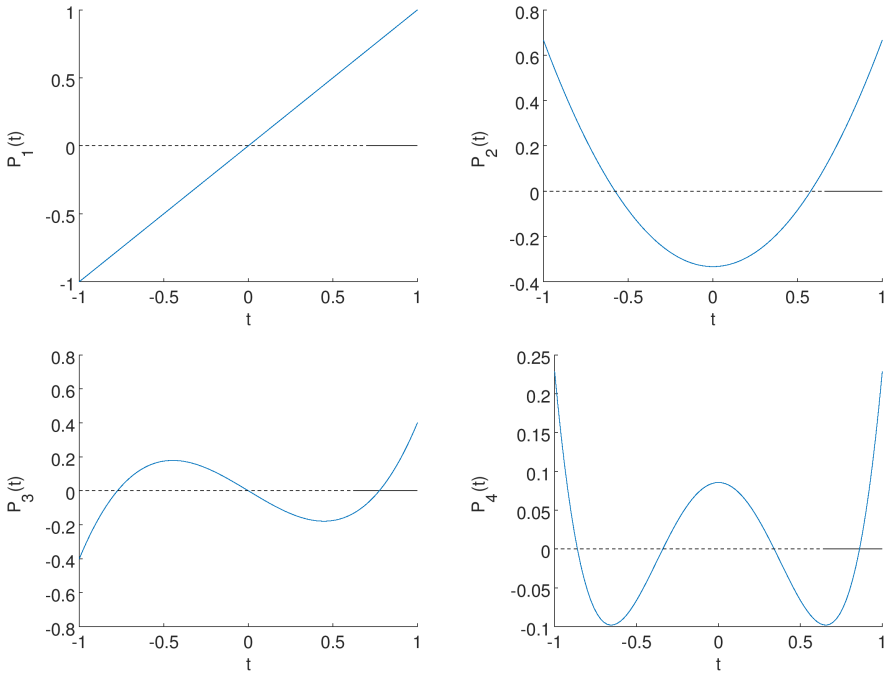
This function evaluates a single value $P_n(t)$ of the monic Legendre polynomial P_n of degree n, at evaluation point t with $|t| \leq 1$. Here $P_0 = 1$, $P_1(t) = t$ and P_n is determined by the recurrence

$$P_n(t) = tP_{n-1}(t) - c_n P_{n-2}(t)$$

for $n \geq 2$, where $c_n = (n-1)^2/(4(n-1)^2 - 1)$. Be sure to iterate forward from $n = 0$ rather than recurse backward from n, and do not generate any new function handles. Test that your function gives the right values for small n where you know P_n.

Solution 6 (10 pts)

The code `pleg.m` is embedded in this pdf file. Using script `test_pleg.m`, we draw the 1st to 4th Legendre polynomial computed below. They seem correct.



Problem 7 Implement a MATLAB function `gaussint.m` of the form

```
function [w, t] = gaussint( n )
% n: Number of Gauss weights and points
```

which computes weights w and points t for the n-point Gaussian integration rule

$$\int_{-1}^{1} f(t)dt \approx \sum_{j=1}^{n} w_j f(t_j).$$

(a) Find the points t_j to as high precision as possible, by applying your code `bisection.m` to `pleg.m`. Bracket each t_j initially by the observation that the zeroes of P_{n-1} separate the zeroes of P_n for every n. Thus the single zero of $P_1 = t$ separates the interval $[-1, 1]$ into two intervals, each containing exactly one zero of P_2. The two zeroes of P_2 separate the interval $[-1, 1]$ into three intervals, and so forth. Thus you will find all the zeroes of P_1, P_2, ..., P_{n-1} in the process of finding all the zeroes of P_n.

(b) Find the weights w_j to as high precision as possible by applying your code `gadap.m` to

$$w_j = \int_{-1}^{1} L_j(t)^2 dt$$

where L_j is the jth Lagrange basis polynomial for interpolating at $t_1, t_2, \ldots, t_n$.

(c) For $1 \leq n \leq 20$, test that your weights and points integrate monomials $f(t) = t^j$ exactly for $0 \leq j \leq 2n - 1$.

Solution 7 (15 pts)

(a)

The function `gaussint.m` is embedded. This function uses `bisection.m`, `gadap.m`, `Lj2.m`, which are also embedded. The tables in the solutions below are generated by `test_gaussint.m`.

$\{t_j\}_{j=1}^n$ up to $n = 7$ is give below:

	j=1	2	3	4	5	6	7
n= 1	0.000000						
2	-0.577350	0.577350					
3	-0.774597	0.000000	0.774597				
4	-0.861136	-0.339981	0.339981	0.861136			
5	-0.906180	-0.538469	0.000000	0.538469	0.906180		
6	-0.932470	-0.661209	-0.238619	0.238619	0.661209	0.932470	
7	-0.949108	-0.741531	-0.405845	0.000000	0.405845	0.741531	0.949108

(b)

$\{w_j\}_{j=1}^n$ up to $n = 7$ is give below:

	j=1	2	3	4	5	6	7
n= 1	2.000000						
2	1.000000	1.000000					
3	0.555556	0.888889	0.555556				
4	0.347855	0.652145	0.652145	0.347855			
5	0.236927	0.478629	0.568889	0.478629	0.236927		
6	0.171324	0.360762	0.467914	0.467914	0.360762	0.171324	
7	0.129485	0.279705	0.381830	0.417959	0.381830	0.279705	0.129485

(c)

The results are shown below. Here n is the number of Gauss points; p is the degree of the polynomial t^p we are integrating; A is the analytic result; e is the absolute error of our quadrature results.

n	p	A	e
1	0	2.000000	0.000000e+000
1	1	0.000000	0.000000e+000
2	2	0.666667	3.996803e-014
2	3	0.000000	0.000000e+000
3	4	0.400000	4.996004e-014
3	5	0.000000	0.000000e+000
4	6	0.285714	8.926193e-014
4	7	0.000000	0.000000e+000
5	8	0.222222	2.339795e-014
5	9	0.000000	0.000000e+000
6	10	0.181818	3.586020e-014
6	11	0.000000	2.949030e-017
7	12	0.153846	1.234013e-013
7	13	0.000000	1.092876e-016