Return this problem set as the first part of your solution set.

Name: __________________________________________

1. Starting from the generating function for Legendre polynomials
\[ h(x,t) = \frac{1}{\sqrt{1 - 2xt + t^2}} = \sum_{n=0}^{\infty} P_n(x)t^n, \] (1)
derive:
   a. The three term recursion formula for Legendre polynomials, and
   b. The relation \( xP_n'(x) = nP_n(x) + P_{n-1}'(x) \).

Hint: Note that \( h(x,t) \) satisfies \( (1 - 2xt + t^2) \frac{\partial h}{\partial t} = (x - t)h \), with a similar relation between \( \frac{\partial h}{\partial t} \) and \( \frac{\partial h}{\partial x} \).

Solution:
   a. Applying the relation \((1 - 2xt + t^2) \frac{\partial h}{\partial t} = (x - t)h\) to the RHS of (1) gives
   \[
   \sum nP_n(x)t^n - 2x \sum nP_n(x)t^n + \sumnP_n(x)t^{n+1} = \sum xP_n(x)t^n - \sum P_n(x)t^{n+1}. \]
After shifting the summation indices so they in each sum give rise to the power \( t^n \), equating coefficients gives the recursion relation \( (n+1)P_{n+1}(x) - 2nP_n(x) + (n-1)P_{n-1}(x) = xP_n(x) - P_{n-1}(x) = 0 \), which readily simplifies to the standard form \( (n+1)P_{n+1}(x) - (2n+1)xP_n(x) + nP_{n-1}(x) = 0 \).

   b. The relation \( t \frac{\partial h}{\partial t} = (x - t) \frac{\partial h}{\partial x} \) gives
   \[
   \sum nP_n(x)t^n = (x - t) \sum P_n'(x)t^n = \sum xP_n'(x)t^n - \sum P_{n-1}'(x)t^n. \]
After which equating coefficients gives the desired relation.

Comment: Differentiating the three term relation and then applying the result in Part (b) (to eliminate the \( xP_n'(x) \) term) gives the first derivative recursion \( (2n+1)P_n(x) = \frac{d}{dx}[P_{n+1}(x) - P_{n-1}(x)] \), allowing in turn the remaining standard relations of the Legendre polynomials to be obtained quite readily.

2. Two standard approaches for accelerating slowly converging positive infinite series are given by the formulas by Euler-MacLaurin
\[ \sum_{n=N}^{\infty} f(n) \sim \int_{N}^{\infty} f(x)dx + \frac{1}{2} f(N) + \frac{1}{12} f'(N) + \frac{1}{720} f''(N) - \frac{1}{30240} f^{(5)}(N) + \ldots \] (2)
and by Gregory
\[
\sum_{n=N}^{\infty} f(n) - \int_{N}^{\infty} f(x)dx = \frac{1}{2} f(N) - \frac{1}{12} \Delta f(N) + \frac{1}{24} \Delta^2 f(N) - \frac{19}{720} \Delta^3 f(N) + \frac{3}{160} \Delta^4 f(N) - \ldots \tag{3}
\]

As is most easily shown by using the relations between the linear operators \( I, E, D \), we can pick up the Euler-MacLaurin coefficients from the Taylor expansion

\[
\left\{ \frac{1 + x}{x} \right\} = \frac{1}{2} - \frac{x}{12} + \frac{x^2}{720} - \frac{x^3}{30240} + - \ldots
\]

Similarly, relate \( I, E, D \) and \( \Delta \) to determine the function from which we can pick up the Gregory coefficients

\[
\sum_{n=N}^{\infty} f(n) - \int_{N}^{\infty} f(x)dx = \left( \frac{I}{I - E} + \frac{I}{D} \right) f(N).
\]

**Hint:** In both cases, the key first step is to show that \( \sum_{n=0}^{\infty} f(n) - \int_{0}^{\infty} f(x)dx = \left( \frac{I}{I - E} + \frac{I}{D} \right) f(N) \).

**Solution:**

For both the Euler-MacLaurin and the Gregory derivations, we first note that

\[
\sum_{n=N}^{\infty} f(n) = (I + E + E^2 + \cdots) f(N) = \frac{I}{I - E} f(N) \quad \text{and that} \quad D \int_{N}^{\infty} f(x)dx = -f(N), \quad \text{implying}
\]

\[
\int_{N}^{\infty} f(x)dx = -\frac{I}{D} f(N). \quad \text{The difference becomes} \quad \sum_{n=1}^{\infty} \left( \frac{I}{I - E} + \frac{I}{D} \right) f(N). \quad \text{In the Gregory case, we want to express both} \ E \ \text{and} \ D \ \text{in terms of} \ \Delta. \ \text{Noting that} \ \Delta = E - I \ \text{and} \ e^\Delta = E = I + \Delta, \ \text{i.e.} \ D = \log(I + \Delta), \ \text{we get}
\]

\[
\frac{I}{I - E} + \frac{I}{D} = -\frac{I}{\Delta} + \frac{I}{\log(I + \Delta)}, \quad \text{telling that the function becomes} \quad -\frac{1}{x} + \frac{1}{\log(1 + x)}.
\]

3. We want to converge to the root \( x = 0 \) of \( f(x) = x - \sin x \) by iteration. If we start with \( x_0 = 1 \), estimate roughly how many iterations it will take to reach an error of \( 10^{-6} \) when using

a. \( x_{n+1} = \sin x_n \),

b. \( x_{n+1} = x_n - f(x_n)/f'(x_n) \).

**Hint:** For both (a) and (b), you may find it helpful to note that \( \sin x \approx x - \frac{1}{6} x^3 \) for \( x \) small.

**Solution:**

a. \( x_{n+1} = \sin x_n \approx x_n - \frac{1}{6} x_n^3 \Rightarrow x_{n+1} - x_n \approx -\frac{1}{6} x_n^3 \). This last relation is a good approximation to the ODE

\[
\frac{d}{dn} x(n) = -\frac{1}{6} x(n)^3 \quad \text{with solution} \quad x(n) = \sqrt{3/(n + c)} \quad \text{where} \ c \ \text{is some constant. This evaluates to}
\]

\( 10^{-6} \) \text{ when} \ n \approx 3 \cdot 10^{12}.

b. With \( f(x) = x - \sin x \), \( f'(x) = 1 - \cos x \), the Newton iteration becomes

\[
x_{n+1} = x_n - \frac{x_n - \sin x_n}{1 - \cos x_n} \approx x_n - \frac{x_n^3/6}{1 - \cos x_n/2} = x_n - \frac{1}{2} x_n = \frac{1}{2} x_n.
\]

The convergence will be like that of a geometric progression with the ratio 2/3. For a rough estimate, we note that 2 iterations gains a factor of 4/9, which is slightly better than 1/2. Knowing that \( 2^{10} \approx 10^3 \), we deduce that 40 iterations will easily suffice.
Comment: Having observed that \( f(x) \) has a triple root at \( x = 0 \), a much better strategy than either of the present iteration schemes would be to use \( x_{n+1} = x_n - \frac{3 f(x_n)}{f'(x_n)} \), i.e. to restore quadratic convergence in the Newton approach by increasing the amount of update by a factor of three.

4. Construct an infinitely differentiable function \( f(x) \) (in simple, closed form) such that the fixed point iteration \( x_{n+1} = f(x_n) \) for any start point \( x_0 \neq 0 \) eventually hops between +1 and -1. Justify that your proposed function satisfies the requirements.

Solution:
A function \( f(x) \) schematically looking as shown below will satisfy the requirements:

It will clearly hold that \( x_{n+1} \) will be of opposite sign to \( x_n \), and the iterates will increase monotonically in magnitude towards one if started with \( |x_0| < 1 \), else decrease monotonically in magnitude towards one. A simple example of a function of this type (recalling that we need \( f(-1) = +1 \) and \( f(+1) = -1 \)) is \( f(x) = -\frac{4}{\pi} \arctan(x) \).

5. Consider quadrature approximations of the form

\[
\int_{-1}^{1} f(x)dx \approx \{w_{-1}f(-1) + w_0f(0) + w_1f(+1)\} + \{u_{-1}f'(-1) + u_1f'(+1)\}.
\]

a. What values for the weights \( w_{-1}, w_0, w_1 \) and \( u_{-1}, u_1 \) are suggested by Euler-MacLaurin’s formula (cf. Problem 2 above)?

b. Are the weights obtained in Part (a) above optimal? Either justify why they are, or produce a different set of weights that makes the formula exact for still higher degree polynomials.

Solution:
Not worrying about convergence/divergence, we obtain from (2)

\[
\begin{align*}
f(-1) + f(0) + f(1) + f(2) + \ldots &= \int_{-1}^{\infty} f(x)dx + \frac{1}{12} f(-1) + \frac{1}{72} f'(-1) + \ldots \\
f(1) + f(2) + \ldots &= \int_{1}^{\infty} f(x)dx + \frac{1}{12} f(+1) + \frac{1}{72} f'(+1) + \ldots
\end{align*}
\]

Subtraction and slight rearrangement gives from this the weight set \( w_{-1} = \frac{1}{2}, w_0 = 1, w_1 = \frac{1}{2} \);
\( u_{-1} = \frac{1}{12}, u_1 = -\frac{1}{12} \).
b. By symmetries, the weights above give the exact result zero for all odd functions. Testing shows the formula to be exact for \( f \equiv 1 \) and \( f \equiv x^2 \), but that it fails already for \( f \equiv x^4 \). Thus, we set out to find a weight set that does better. Keeping the symmetries we have just observed, let the new quadrature formula be

\[
\int_{-1}^{1} f(x) \, dx \approx \{\alpha f(-1) + \beta f(0) + \alpha f(+1)\} + \{\gamma f'(-1) - \gamma f'(+1)\}.
\]

This will again be exact for all odd functions. We have three free parameters, so we should be able to force it to be exact also for \( f \equiv x^4 \). Applying it in turn to \( f(x) = 1, x^2, x^4 \) gives

\[
\begin{align*}
2 &= \int_{-1}^{1} dx = 2\alpha + \beta \\
\frac{2}{3} &= \int_{-1}^{1} x^2 \, dx = 2\alpha - 4\gamma \\
\frac{2}{5} &= \int_{-1}^{1} x^4 \, dx = 2\alpha - 8\gamma 
\end{align*}
\]

with the solution \( \alpha = \frac{7}{15}, \beta = \frac{16}{15}, \gamma = \frac{1}{15} \).

6. **Multiple choice** - Mark below, in the boxes to the right of each question, either True or False (i.e. not always correct). You do not need to give any explanations for your answers to this problem.

<table>
<thead>
<tr>
<th></th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Horner’s scheme is particularly well suited for evaluating polynomials in the form that is obtained from Newton’s interpolation formula</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>b. Hermite’s interpolation formula for data ( (x_i, f(x_i), f'(x_i)), i = 1, 2, \ldots, n ), uses two sets of basis functions, ( h_i(x) ) and ( \tilde{h}_i(x) ). It holds that ( h_j'(x_j) = \tilde{h}_i(x_j) = 0 ) for ( j = 1, 2, \ldots, n ).</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>c. With the standard definition of the linear difference operators, it holds that ( I + E \nabla = E ).</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>d. The tridiagonal linear system that arises from obtaining the coefficients in a cubic spline is always diagonally dominant, i.e. the Thomas’ algorithm is well suited for the task of solving it.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>e. A cubic spline over the nodes ( x_i, i = 1, 2, \ldots, n ) is commonly expressed as a linear combination of ( n ) ( B )-splines, one centered at each node point.</td>
<td>☐</td>
<td>☒</td>
</tr>
<tr>
<td>f. For a cubic spline, the second derivative may jump in value between two adjacent subintervals.</td>
<td>☐</td>
<td>☒</td>
</tr>
<tr>
<td>g. A spline interpolant (of any order ( 1, 2, 3, \ldots )) to cardinal data (one at one node point, and zero at all other node points) will always be oscillatory far out in both directions.</td>
<td>☐</td>
<td>☒</td>
</tr>
<tr>
<td>h. The ( n^{th} ) root unity ( \omega ) is customarily defined as ( \omega = e^{\pi i/N} ).</td>
<td>☐</td>
<td>☒</td>
</tr>
</tbody>
</table>
i. Evaluation of \[
\begin{bmatrix}
 z_0 & z_{N-1} & z_{N-2} & \cdots & z_1 \\
 z_1 & z_0 & z_{N-1} & \cdots & z_2 \\
 \vdots & \vdots & \vdots & \ddots & \vdots \\
 \vdots & \vdots & \vdots & \ddots & \vdots \\
 z_{N-1} & z_{N-2} & \cdots & \cdots & z_0 \\
\end{bmatrix}
\begin{bmatrix}
 x_0 \\
x_1 \\
 \vdots \\
 \vdots \\
x_{N-1} \\
\end{bmatrix}
\]
requires \( O(N^2) \) arithmetic operations.

j. When using bisection to locate a simple zero of a scalar function, we can be certain to reach an accuracy very close to machine precision \( 10^{-16} \).

k. If \( g(x) \) is a continuous function and \( x_{n+1} = g(x_n) \) has more than one fixed point, then iterations cannot be locally convergent to each of the fixed points.

l. If the error in an iterative scheme decreases like \( O(2^{-n}) \), the scheme is ‘quadratically’ convergent.

m. Local convergence/divergence rates for fixed point iteration can be analyzed by considering the determinant of the Jacobian matrix corresponding to the system’s right hand side.

n. A main approach for proving Weierstrass’ theorem on polynomial approximation starts by analyzing some main properties of Bernoulli polynomials.

o. The IMT method for numerical quadrature combines Gaussian quadrature with a change of variables.

p. The standard normalization for Legendre polynomials is to require that \( \int_{-1}^{1} P_n(x)^2 \, dx = 1 \), for \( n = 0,1,2,3,... \).

q. The highest order term in the Chebyshev polynomial \( T_n(x) \) is \( 2^{n-1} x^n \).

r. The Chebyshev coefficients of a function can be approximated rapidly with use of the FST (Fast Sine Transform).

s. The Second algorithm of Ramez converges quadratically fast to the optimal polynomial approximation of a continuous function.

t. The 6-node Gaussian quadrature formula for \( \int_{-1}^{1} f(x) \, dx \) is exact for all polynomials up through degree 11.

u. There is an ‘Euler-MacLaurin-like’ formula available for summation of infinite series with terms of alternating signs. Unlike regular Euler-MacLaurin, it gives an infinite sum \( \sum_{n=N}^{\infty} (-1)^n f(n) \) in terms of \( f \) and odd derivatives of \( f \) evaluated at \( N \), without any integral present.

v. The cost for solving an \( n \times n \) five-diagonal linear system (when pivoting system is needed) becomes \( O(n^2) \) operations.
w. It holds that
\[
\begin{bmatrix}
1 & a_2 & a_3 & a_4 \\
a_2 & 1 & b_3 & b_4 \\
a_3 & b_3 & 1 & c_4 \\
a_4 & b_4 & c_4 & 1
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 1 \\
1 & b_3 & \tau \\
1 & c_4 & \tau \\
\tau & \tau & \tau
\end{bmatrix} = \begin{bmatrix}
1 & a_2 & a_3 & a_4 \\
a_2 & 1 & b_3 & b_4 \\
a_3 & b_3 & 1 & c_4 \\
a_4 & b_4 & c_4 & 1
\end{bmatrix}^{-1} \cdot \begin{bmatrix}
1 & a_2 & a_3 & a_4 \\
a_2 & 1 & b_3 & b_4 \\
a_3 & b_3 & 1 & c_4 \\
a_4 & b_4 & c_4 & 1
\end{bmatrix}^{-1}.
\]

x. To leading order, the cost is the same for unpivoted Gaussian elimination (not recommended), as when using partial pivoting.

y. \( \text{cond}(A) \) (as defined for solving linear systems) satisfies \( \text{cond}(A) \geq 1 \), with equality when \( A \) is a diagonal matrix.

z. The computationally preferred method for finding Gaussian quadrature nodes and weights contains as a key step the calculation of the eigenvalues of a symmetric tridiagonal matrix.