Solution to MA5315 Midsemester Exam

1. Programming

   - Write a MATLAB function that accepts an integer $N$ and returns

     \[ S = \frac{1}{1^3} + \frac{1}{2^3} + \ldots + \frac{1}{N^3} \]

     On entry to the function, check whether $N$ is positive and quit with an error message if it is not. Compute the sum with a for loop. Be sure that any indentations you make are clearly visible.

     **Solution** The program might look like this:

     ```matlab
     function S = midsem(N)
     if N <= 0
       error('N must be positive')
     end
     S = 0;
     for n = 1:N
       S = S + 1/n^3;
     end
     end % midsem
     ```

     Indentation of the body of the function is optional. Indentation of the body of the if and for constructs is mandatory. A semicolon is optional after error and the final end statement is optional.

2. Finite Precision—answer one question

   (a) Many programs provide only a pure absolute error control. Suppose that you want to compute a quantity that you know is about $10^{5}$. Is it meaningful to ask the code for an accuracy of $10^{-10}$? What about $10^{-12}$? (Your answer should be for IEEE double precision arithmetic.) If a tolerance is not meaningful, explain why.

     **Solution** An absolute error tolerance of $\tau$ corresponds to a relative error of

     \[ \left| \frac{true - approx}{true} \right| \leq \frac{\tau}{|true|} \]
It is not meaningful to ask for a relative accuracy less than a unit roundoff because that is the best we can expect of the floating point representation of the true value. In the IEEE standard, the unit roundoff, called \texttt{eps} in MATLAB, is about $10^{-16}$. Accordingly, if \textit{true} is about $10^5$, an absolute error tolerance of $10^{-12}$ corresponds to a relative accuracy of about $10^{-12}/10^5 = 10^{-17}$. This is smaller than a unit roundoff, so this tolerance is not meaningful in the precision available. On the other hand, an absolute error tolerance of $10^{-10}$ corresponds to a relative error of about $10^{-15}$, which is very stringent, but still meaningful because it is bigger than a unit roundoff.

(b) Explain why evaluating \[ \frac{1 - \cos(x^2)}{x} \] in a straightforward way is unsatisfactory for small \( x \). You may find the series \[ \cos(x) = 1 - \frac{x^2}{2} + \frac{x^4}{24} - \ldots \] useful in this explanation. You are asked to explain the difficulties, not resolve them.

\textbf{Solution} The series for \( \cos(x) \) shows that \[ \cos(x^2) = 1 - \frac{x^4}{2} + \frac{x^8}{24} - \ldots \] Obviously \( \cos(x^2) \) approaches 1 very quickly as \( x \to 0 \). As a result, there is severe cancelation in forming the numerator of the fraction, making any error in evaluating \( \cos(x^2) \) relatively large in the difference. The denominator of the fraction is small, which amplifies the error in the numerator.

3. \textbf{Nonlinear Equations}—answer two questions

(a) Explain what it means for \( \alpha \) to be a root of \( f(x) \) of multiplicity \( m \). Show that if \( m \) is even, then \( \alpha \) is a root of \( f'(x) \) of odd multiplicity.

\textbf{Solution} \( \alpha \) is a root of multiplicity \( m \) if \( f(x) = (x - \alpha)^m g(x) \) and \( g(\alpha) \neq 0 \). Differentiating the expression we have

\[
\begin{align*}
f'(x) &= m(x - \alpha)^{m-1} g(x) + (x - \alpha)^m g'(x) \\
&= (x - \alpha)^{m-1} \left[m g(x) + (x - \alpha) g'(x)\right] \\
&= (x - \alpha)^{m-1} G(x)
\end{align*}
\]

Now \( G(\alpha) = m g(\alpha) \neq 0 \), so by definition, \( \alpha \) is a root of \( f'(x) \) of multiplicity \( m - 1 \).

(b) For solving \( f(x) = 0 \), the method of bisection has both good and bad points. State two points of each kind.

\textbf{Solution} Important good points:
• The rate of convergence is not affected by the number and multiplicity of roots.
• The function need not be smooth.
• It is stable at limiting precision.
• It is globally convergent.

Important bad points:
• It can’t calculate a root of even multiplicity except by accident.
• There are methods that are much faster for simple roots and smooth \( f(x) \).
• It doesn’t generalize to functions of several variables, including one complex variable.
• It might compute a pole.
• You can’t be sure of getting a particular root.

(c) State Newton’s method for finding a root \( \alpha \) of \( f(x) = 0 \). It is often said that the method is quadratically convergent. What does this mean? Newton’s method may not behave like a quadratically convergent method—state one situation for which this is the case.

**Solution** Newton’s method improves an iterate \( x_n \) by

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}
\]

A method is said to be quadratically convergent to a root \( \alpha \) if the iterates \( \{x_n\} \) satisfy

\[
\lim_{n \to \infty} \frac{x_{n+1} - \alpha}{(x_n - \alpha)^2} = c \neq 0
\]

or equivalently, \( x_{n+1} - \alpha \approx c(x_n - \alpha)^2 \) for “large” \( n \). The method does not behave like a quadratically convergent method when

• The root \( \alpha \) is a multiple root.
• Iterates \( x_n \) are too far from \( \alpha \) for a local analysis to apply. And, when a local analysis is partially applicable, a cluster of simple roots leads to the behavior expected for a multiple root.
• At limiting precision the function values vary erratically in magnitude and sign, so an analysis based on continuous functions does not apply. As limiting precision is approached, errors in evaluating \( f(x_n) \) can lead to behavior that does not appear to be quadratic convergence.

4. Interpolation—answer one question

(a) Recall that if \( F(x) \) is smooth enough and \( P_3(x) \) interpolates \( F(x) \) at three distinct nodes \( x_i \), then for any given \( x \), there is a point \( \xi \) such
that

\[ F(x) - P_3(x) = \frac{F^{(3)}(\xi)}{3!} \prod_{i=1}^{3} (x - x_i) = \frac{F^{(3)}(\xi)}{3!} \omega(x) \quad (1) \]

i. Write down the Lagrangian form of \( P_3(x) \).

ii. Consider a function \( f(x) \) defined by the table

\[
x = [0.005 \ 0.010 \ 0.015 \ 0.020 \ 0.030]; \\
f = [200.0010 \ 100.0020 \ 66.6692 \ 50.0033 \ 33.3383];
\]

A. If you want to approximate \( f(0.012) \) by interpolating at three nodes, some choices are better than others. What would be a good choice of nodes? Why?

B. If you want to approximate \( f(0.1) \) by interpolating at three nodes, which nodes would you use? Why should you be concerned about the accuracy of this approximation?

**Solution**

i. The quadratic polynomial is

\[
P_3(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} F(x_1) + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} F(x_2) + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} F(x_3)
\]

ii. For approximating \( f(x) \) by quadratic interpolation

A. Choosing nodes as close as possible to the given \( x \) makes the factor \( \omega(x) \) in the error expression (1) relatively small. Here it would be best to use the nodes \{0.005, 0.10, 0.015\} because they are the three nodes closest to \( x = 0.012 \).

B. It would be best to use the nodes \{0.015, 0.020, 0.030\} because they are closest to the given \( x \) for the same reason as in the previous question. However, the factor \( \omega(x) \) grows rapidly as \( x \) moves away from the span of the data, extrapolation. In this case the given \( x = 0.1 \) is well outside the span of the data and correspondingly, the factor \( \omega(0.1) \) in the error expression (1) is relatively large.

(b) We have considered several kinds of interpolatory splines for approximating a smooth function \( f(x) \) at nodes \( x_1 < x_2 < \ldots < x_N \). One kind is implemented in `pchip` and another in `spline`. Let \( h = \max_i |x_{i+1} - x_i| \).

i. What properties define the spline \( S(x) \) implemented in `spline`?

The default end condition is not-a-knot and optionally it is the end condition for a complete cubic spline. You are to explain *one*—for this exam you are to consider the two possibilities as being equivalent.
ii. How does the spline $H(x)$ of \texttt{pchip} differ from the spline $S(x)$ of \texttt{spline}? Specifically, how do they differ as regards
A. smoothness of the spline,
B. accuracy of the spline as $h \to 0$, and
C. preservation of monotonicity in the data?

\textbf{Solution}

i. $S(x)$ is defined by the properties
- $S(x)$ is a cubic polynomial on each $[x_i, x_{i+1}]$
- $S(x_i) = f(x_i)$ for each $x_i$
- $S(x) \in C^2[x_1, x_N]$
- $S(x)$ satisfies an end condition at $x_1$ and $x_N$. The form of the condition is the same at both ends. At $x_1$ it is
  - For a complete cubic spline, $S'(x_1) = f'(x_1)$
  - The not-a-knot condition is that $S^{(3)}(x)$ is continuous at $x_2$. Equivalently, the cubic polynomial on $[x_1, x_2]$ is the same as the one on $[x_2, x_3]$.

ii. The splines differ in the following ways
A. $S(x)$ is smoother. Specifically, $S(x) \in C^2[x_1, x_N]$ and $H(x) \in C^1[x_1, x_N]$.
B. $S(x)$ converges faster to $f(x)$ as $h \to 0$.
C. A spline is said to preserve monotonicity in the data if it is increasing on $(x_i, x_{i+1})$ whenever $f(x_i) < f(x_{i+1})$ and decreasing whenever $f(x_i) > f(x_{i+1})$. $H(x)$ preserves monotonicity in the data for any mesh. Because $S'(x) \to f'(x)$ as $h \to 0$, this spline preserves monotonicity if $h$ is sufficiently small, but it may not preserve monotonicity for “large” $h$. 

5