1. Solve the system

\[ \begin{align*}
x_1 + x_2 &= 2 \\
\epsilon x_1 + x_2 &= 1
\end{align*} \]

for \( x_1 \) and \( x_2 \) in terms of \( \epsilon \). If this were done on a computer, why would the solutions become erroneous as \( \epsilon \to 0 \)?

2. Suppose you required a numerical quantity which could be computed either directly or, in a more elegant way, using recursion. Would it be better to write the program using the direct computation (perhaps involving loops) or write it using recursive function calls? Give reasons for your answer.

3. In solving linear equations, when is it advisable to do an LU factorization of the coefficient matrix \( A \).

4. Let \( f \) be a function with the following \( C^k \)-norms on \( [0, 1] \):

\[
\begin{align*}
\max |f'(x)| &= 35.65, \quad \max |f''(x)| = 3505.85 \\
\max |f'''(x)| &= 65601.05 \quad \text{and} \quad \max |f''''(x)| = 35608345601.05
\end{align*}
\]

What numerical integration technique would you recommend for finding \( \int_0^1 f(x)dx \) and why?

5. Assume that we want to solve the matrix equation \( AX = B \) with a small relative error. Here \( A \) is a non-singular \( n \times n \) matrix \((a_{ij})\), \( X \) is the column vector of unknowns \(< x_1, x_2, x_3, \ldots, x_n >\) and \( B \) is the column vector of values \(< b_1, b_2, b_3, \ldots, b_n >\), with \( \|B\|_\infty = 10.2 \). We have computed an approximate solution, \( X_{\text{computed}} \), for which the residual is quite small, i.e. \( \|B - AX_{\text{computed}}\|_\infty = 0.000001 \).

5a. First suppose that \( \|A\| = 35.6 \) and \( \|A^{-1}\| = 3068.8 \). Can we expect Gaussian elimination with scaled partial pivoting to give us a good approximate solution? Defend your answer.

5b. Next suppose that \( \|A\| = 35.6 \) and \( \|A^{-1}\| = 308432980345.8 \). Can we expect Gaussian elimination with scaled partial pivoting to give us a good approximate solution? Defend your answer.

6. Let \( f(x) = x(4 - x) \) and let \( P \) be the partition \( \{0, 1, 2, 3, 4\} \) of \( [0, 4] \).

6a. Estimate \( \int_0^4 f(x)dx \) by averaging the upper sum \( U(f, P) \) and the lower sum \( L(f, P) \).

6b. Use the trapezoidal rule \( T(f, P) \) to estimate \( \int_0^4 f(x)dx \). Is \( T(f, P) \) an overestimation or an underestimation? Why?

7. Use either Newton’s interpolation method or a table of divided differences to find the polynomial \( p(x) \) which satisfies \( p(-1) = 8, p(0) = 4, p(1) = 6, \) and \( p(2) = 20 \). Note that one way requires fewer calculations. You may leave \( p(x) \) in Newton’s form \((a_0 + a_1(x+1) + a_2(x+1)(x+2) \ldots) \) and not multiply it out.

9. For a function \( f \), it is known that all of its roots are double roots, i.e. if \( f(x) = 0 \) then \( f'(x) = 0 \). What would be the safest numerical technique for finding a root of \( f \) in \([0, 1]\) and why?

10. Is it possible to represent the number \(-2047/2048\) in single precision? If so, do it; if not, explain why not.

11. A careless programmer has put the following statements in a program. Why will the variable \( y \) most likely be inaccurate? What should the programmer have done to ensure an accurate value for the variable \( y \)?
float x, y;
x = 0.00000000000005;
y = sin(x) - x;

12. One approach to speed up solving linear systems of equations (on a multiple-core system) is to use a parallel algorithm for Gaussian elimination and implement this with a multi-threaded program. For example, the main() thread might only find the pivot row and do the swap. Then each of \( n \) auxiliary threads will work on (roughly) \( 1/n \)-th of the augmented matrix. What are the key issues involved here?

13. A programmer codes a multi-threaded C program to solve linear systems of equations using from 1 to 7 auxiliary threads (besides main()). Four times, he runs and times the program (real time) on an 8-cpu system using from 1 to 7 auxiliary threads and obtains the following results:

<table>
<thead>
<tr>
<th>Aux. Thr.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall clock:</td>
<td>05.84</td>
<td>03.82</td>
<td>03.45</td>
<td>03.14</td>
<td>03.39</td>
<td>03.26</td>
<td>02.88</td>
</tr>
<tr>
<td>Wall clock:</td>
<td>04.67</td>
<td>03.53</td>
<td>03.49</td>
<td>03.15</td>
<td>02.95</td>
<td>03.01</td>
<td>02.93</td>
</tr>
<tr>
<td>Wall clock:</td>
<td>05.35</td>
<td>03.78</td>
<td>03.56</td>
<td>03.93</td>
<td>03.18</td>
<td>02.95</td>
<td>03.14</td>
</tr>
<tr>
<td>Wall clock:</td>
<td>05.33</td>
<td>03.28</td>
<td>03.20</td>
<td>03.57</td>
<td>03.34</td>
<td>03.05</td>
<td>03.20</td>
</tr>
</tbody>
</table>

The programmer is puzzled that he did not achieve a speedup of seven times using the full number of threads. Explain.