Math 174 Final
December 13, 2013

- Please put your name, ID number, and sign and date.
- There are 8 problems worth a total of 200 points.
- You must show your work to receive credit.

Print Name: ___________________________________________

Student ID: ___________________________________________

Signature and Date: ___________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/25</td>
</tr>
<tr>
<td>2</td>
<td>/25</td>
</tr>
<tr>
<td>3</td>
<td>/25</td>
</tr>
<tr>
<td>4</td>
<td>/25</td>
</tr>
<tr>
<td>5</td>
<td>/25</td>
</tr>
<tr>
<td>6</td>
<td>/25</td>
</tr>
<tr>
<td>7</td>
<td>/25</td>
</tr>
<tr>
<td>8</td>
<td>/25</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>/200</td>
</tr>
</tbody>
</table>

1
1. (25 pts) Given the following header for a Matlab function:

function [x] = SolvePLU(P,L,U,b,n)

where \( P, L, U \) make up the PLU factorization for an \( n \times n \) matrix \( A \) (\( PA = LU \)) and \( b \) is a vector, complete the function so that it uses the PLU factorization to solve the linear system \( Ax = b \) for \( x \). Do not use Matlab’s in-built inverse or matrix-matrix or matrix-vector multiplications.

function [x] = SolvePLU(P,L,U,b,n)

for i = 1:n
   c(i,1) = 0;
   for j = 1:n
      c(i,1) = c(i,1)+P(i,j)*b(j,1);
   end
end

for i = 1:n
   thesum = 0;
   for j = 1:i-1
      thesum = thesum+L(i,j)*c(j,1);
   end
   y(i,1) = b(i,1)-thesum;
end

for i = n:-1:1
   thesum = 0;
   for j = i+1:n
      thesum = thesum+U(i,j)*y(j,1);
   end
   x(i,1) = (b(i,1)-thesum)/U(i,i);
end

end
2. (25 pts) **Circle** the best answer in each part. You do **not** have to show your work in this problem.

(a) The number of **flops** needed to perform LU factorization on an \( n \times n \) matrix is:

\[
\begin{align*}
(i) \ O(n) & \\
(ii) \ O(n^2) & \\
(iii) \ O(n^3) & \\
(iv) \ O(n^4) &
\end{align*}
\]

(b) Suppose runtime is mainly influenced by flops. Let \( A \) be an \( n \times n \) matrix and \( B \) a \( 2n \times 2n \) matrix, both **tridiagonal** and with nonzero diagonal elements. Then each iteration of Gauss-Seidel when solving \( By = c \) is this many **times slower** than when solving \( Ax = b \):

\[
\begin{align*}
(i) \ 2 & \\
(ii) \ 3 & \\
(iii) \ 4 & \\
(iv) \ 8 &
\end{align*}
\]

(c) Suppose \( f \) is a smooth function and \( x_0 \) is a location. Let \( h_1, h_2 > 0 \) be two small stepsizes with \( h_1 = 2h_2 \). The absolute error of second order central differencing approximating \( f'(x_0) \) when using \( h_2 \) is this many **times smaller** than when using \( h_1 \):

\[
\begin{align*}
(i) \ 2 & \\
(ii) \ 3 & \\
(iii) \ 4 & \\
(iv) \ 8 &
\end{align*}
\]

(d) Let \( p(x) \) be a polynomial of degree 3 and let \( x_0, x_1, x_2, x_3, x_4 \) be **five** distinct nodes. Then the interpolating polynomial of least degree for the data points \( (x_0, p(x_0)), (x_1, p(x_1)), (x_2, p(x_2)), (x_3, p(x_3)), (x_4, p(x_4)) \) has this **degree**:

\[
\begin{align*}
(i) \ 3 & \\
(ii) \ 4 & \\
(iii) \ 5 & \\
(iv) \ \text{none of the above} &
\end{align*}
\]

(e) Let \( f \) be a smooth function with **two** roots and \( x_0 \) an initial guess. If Newton’s method converges, it will always converge to the root that is **closest** to \( x_0 \).

**TRUE** or **FALSE**

(f) Gauss-Seidel and Jacobi methods’ sequences of approximations are exactly the **same** when using the same initial guess and applied to the same linear system \( Ax = b \), where \( A \) is **upper triangular** with nonzero diagonal elements.

**TRUE** or **FALSE**

(g) Trapezoid rule on a **concave down** function gives an **underestimate** of the real integral.

**TRUE** or **FALSE**

(h) Midpoint rule on a **concave down** function gives an **underestimate** of the real integral.

**TRUE** or **FALSE**
3. (25 pts) For each part below, find examples of the specified quantities. Be sure to justify that your choices satisfy the listed requirements.

(a) Let \( f(x) = x^2 - 4 \). Come up with a starting interval \([a_0, b_0]\) such that \([a_2, b_2]\) generated by bisection method satisfies \( a_2 \neq a_0, b_2 \neq b_0 \).

\([0.9, 2.9]\)

(b) Come up with a sequence that converges to 5 with order of convergence 3.

\[ x_n = 5 + 2^{-3^n} \]
(c) Come up with two different polynomials of any degree that interpolate the following data:
\[
\begin{array}{c|ccc}
  x & -1 & 0 & 1 \\
  y & 1 & 2 & -1 \\
\end{array}
\]

\[p(x)\] the interpolating polynomial for 
\[
\begin{array}{c|ccc}
  x & -1 & 0 & 2 \\
  y & 1 & -1 & 0 \\
\end{array}
\]
and \[q(x)\] the interpolating polynomial for 
\[
\begin{array}{c|ccc}
  x & -1 & 0 & 1 \\
  y & 1 & 2 & -1 \\
\end{array}
\].

(d) Come up with three data points with distinct nodes such that the interpolating polynomial of least degree has degree < 2.

\[
\begin{array}{c|ccc}
  x & 1 & 2 & 3 \\
  y & 0 & 0 & 0 \\
\end{array}
\]
4. (25 pts) Consider the iterative method whose sequence of approximations satisfies:

\[(2D - L)x^{(k+1)} = b + (U + D)x^{(k)},\]

for solving a linear system \(Ax = b\), with \(A = D - L - U\), where \(D\) is diagonal, \(L\) is strictly lower triangular, and \(U\) is strictly upper triangular. When

\[A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},\]

write out the iteration matrix and use it to determine whether this iterative method’s sequence of approximations will converge to the solution of \(Ax = b\) for any initial guess.

\[T = \begin{bmatrix} 2 & 0 \\ 2 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & -1 \\ -1/2 & 3/2 \end{bmatrix} \text{. Eigenvalues satisfy } (1/2 - \lambda)(3/2 - \lambda) - 1/2 = 0 \text{ so } \lambda = \frac{2 \pm \sqrt{3}}{2}.\]

\[\rho(T) = \frac{2 + \sqrt{3}}{2} > 1. \text{ Method does not converge for all initial guesses.}\]
5. (25 pts) Find $a, b, c, d, f$ so that the following is a free or natural ($S''$(endpoints) = 0) cubic spline

\[ S(x) = \begin{cases} 
1 - 2x - 3x^2 + ax^3, & \text{if } -1 \leq x < 2; \\
b + c(x - 2) + d(x - 2)^2 + f(x - 2)^3, & \text{if } 2 \leq x \leq 3
\end{cases} \]

\[
\begin{align*}
-6 - 6a &= 0 \\
2d + 6f &= 0 \\
-15 + 8a &= b \\
-14 + 12a &= c \\
-6 + 12a &= 2d
\end{align*}
\]

so, $a = -1, b = -23, c = -26, d = -9, f = 3$. 

7
6. (25 pts) Consider the table of data:

<table>
<thead>
<tr>
<th>$x$</th>
<th>0.9</th>
<th>1</th>
<th>1.1</th>
<th>1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x)$</td>
<td>0.54</td>
<td>0.5</td>
<td>0.46</td>
<td>0.4</td>
</tr>
</tbody>
</table>

First write out Newton’s form for the interpolating polynomial $p(x)$, then use it to approximate $f'(1)$.

$$p(x) = 0.54 - 0.4(x - 0.9) + (x - 0.9)(x - 1)(x - 1.1)$$

so $f'(1) \approx p'(1) = 0.41$. 
7. (25 pts) Use Taylor series to find the $p$ satisfying
\[
\left| \int_0^h f(x) \, dx - \frac{h}{2} [f(0) + f(h)] \right| = O(h^p).
\]

Taylor series on $F(h)$, where
\[
F(z) = \int_0^z f(x) \, dx,
\]
gives $p = 3$. 
8. (25 pts) Consider

\[ y' = 3(t + 1)y \]

with \( y(0) = 2 \). Use one step of predictor-corrector to compute \( y_1 \approx y(0.1) \) with Midpoint method

\[ y_{i+1} = y_i + hf \left( t_i + \frac{h}{2}, y_i + \frac{h}{2} f(t_i, y_i) \right) \]

as predictor and Trapezoid method

\[ y_{i+1} = y_i + \frac{h}{2} \left( f(t_i, y_i) + f(t_{i+1}, y_{i+1}) \right) \]

as corrector.

\[ \tilde{y}_1 = 2.7245 \text{ so } y_1 = 2.7495. \]