\textbf{P1)}

(a)

(1)

The \( h \), \( u \), and \( v \) vectors were calculated by my MATLAB code. They are written down here:

\( h = [ 6 \ 4 \ 3 \ 4 \ 3 \ 8 ] \)

\( u = [ - 20 \ 13.2 \ 13.318182 \ 12.798635 \ 21.2968 ] \)

\( v_{s1} = [ - 27.35 \ -54.16 \ 12.144091 \ 5.617628 \ -0.164272 ] \)

\( v_{s2} = [ - -5.27 \ -10.896 \ 3.596364 \ 3.339863 \ 1.044636 ] \)

\( z_0 = z_n = 0 \)

Sample 1:

\[
\begin{bmatrix}
20 & 4 & 0 & 0 & 0 \\
4 & 13.2 & 3 & 0 & 0 \\
0 & 3 & 13.318182 & 4 & 0 \\
0 & 0 & 4 & 12.798635 & 3 \\
0 & 0 & 0 & 3 & 21.2968
\end{bmatrix}
\begin{bmatrix}
z_1 \\
z_2 \\
z_3 \\
z_4 \\
z_5
\end{bmatrix}
= 
\begin{bmatrix}
-27.35 \\
-54.16 \\
12.144091 \\
5.617628 \\
-0.164272
\end{bmatrix}
\]

Sample 2:

\[
\begin{bmatrix}
20 & 4 & 0 & 0 & 0 \\
4 & 13.2 & 3 & 0 & 0 \\
0 & 3 & 13.318182 & 4 & 0 \\
0 & 0 & 4 & 12.798635 & 3 \\
0 & 0 & 0 & 3 & 21.2968
\end{bmatrix}
\begin{bmatrix}
z_1 \\
z_2 \\
z_3 \\
z_4 \\
z_5
\end{bmatrix}
= 
\begin{bmatrix}
-5.27 \\
-10.896 \\
3.596364 \\
3.339863 \\
1.044636
\end{bmatrix}
\]
My code uses the nested version of the cubic spline:

\[ y_i = f(t_i) + (x - t_i)\left[C_i + (x - t_i)(B_i + (x - t_i)A_i)\right] \]

For \( t \) between 6 and 10 days:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.270988</td>
<td>1.111768</td>
<td>6.22373996</td>
</tr>
</tbody>
</table>

\[ S_0(7) = 17.33 + (7 - 6)(6.22373996 + (7 - 6)(1.111768 + (7 - 6)(-0.270988))) \]

\[ S_0(7) \approx 24.39451996 \]

From MATLAB: \( S_0(7) \approx 24.3945 \)

(5)

For \( t \) between 6 and 10 days:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.032511</td>
<td>-0.044770</td>
<td>1.394253</td>
</tr>
</tbody>
</table>

\[ S_0(7) = 16.11 + (7 - 6)(1.394253 + (7 - 6)(-0.044770 + (7 - 6)(-0.032511))) \]

\[ S_0(7) \approx 17.426972 \]

From MATLAB: \( S_0(7) \approx 24.4270 \)

(b)

I’m already evaluating the cubic spline in my code at a 100,000 points in order to plot it, so I just used the max() function in MATLAB to get the max of each spline:

\[ \text{Max}_{s_1} = 43.2302 \]

\[ \text{Max}_{s_2} = 19.0561 \]

An alternative way of doing it would be to see where the \( z \) values (equivalent to the second derivative) are negative. You check between the time bounds wherever there is a negative \( z \). It would cut down on the amount of points to evaluate.
(c)

To run the program, run the cubicSplinescript.m with cubic_spline.m and eval_cubic_spline.m is the same directory. I used the code from my homework 3, and I edited to fit the exam problem. Here is the pseudocode from the homework:

splineMat: The matrix that holds the coefficients for the linear/cubic spline

splineMat = cubic_spline(f) {
    f is the values of the data points
    t = [0 6 10 13 17 20 28];
    b, z:= initialize vector of size n
    u, v:= initialize vector of size n whose first value is 0 (unused placeholder)
    initialize a matrix called splineMat with n = 6 rows and 5 columns
    first find the z values by using the algorithm in the Kincaid book on pg. 353
    for i from 1 to n do
        \( A_i = \frac{1}{6h} \)
        \( B_i = \frac{z_i}{2} \)
        \( C_i = -\frac{h}{6}z_{i+1} - \frac{h}{3}z_i + \frac{1}{h}(y_{i+1} - y_i) \)
        splineMat(i,:) = [ Ai, Bi, Ci, lowerbound, upperbound]
    }

y = eval_cubic_spline(splineMat, x, frow) {
    f := each row represents the data points of each sample (i.e. row 1 for sample 1)
    initialize a y vector the length of the x vector
    for j from 1 to length(x) do
        for i from 1 to n do
            find if the x falls between the lower and upper bounds of the cubic polynomial
            if so, evaluate the \( y_j = f(t_i) + (x - t_i)[C_i + (x - t_i)[B_i + (x - t_i)A_i]] \)
            using the A, B, and C values stored in splineMat. Note: \( t_i \) is the lower bound.
        end
    end
}
function [cubic_spline,u,v,h,z] = cubic_spline(f)

Initializations

t = [0 6 10 13 17 20 28]; % n = 6
h = [];
b = [];
u = [0]; % starts off with a placeholder since u_0 is never used
v = [0]; % starts off with a placeholder since v_0 is never used

Cubic Spline

% find the values of z using the algorithm in the book
for i = 1:6
    h = [h t(i+1)-t(i)];
    b = [b 6*(f(i+1) - f(i))/h(i)];
end
u = [u 2*(h(1)+h(2))];
v = [v b(2)-b(1)];
for i = 3:6
    u = [u (2*(h(i)+h(i-1)) - h(i-1)^2/u(i-1))];
    v = [v (b(i)-b(i-1)-h(i-1)*v(i-1)/u(i-1))];
end
z = zeros(7,1);
for i = 6:-1:2
    z(i) = (v(i) - h(i)*z(i+1))/u(i);
end

% find the values of A, B, C that characterize each polynomial in the cubic
% spline

% format: [A, B, C, ti, ti+1]
cubic_spline = [];
for i = 1:6
    A = (1/(6*h(i)))*(z(i+1) - z(i));
    B = z(i)/2;
    C = -h(i)*z(i+1)/6 - h(i)*z(i)/3 + (1/h(i))*(f(i+1) - f(i));
cubic_spline = [cubic_spline; A B C t(i) t(i+1)];
end
function [y] = eval_cubic_spline(splineMat,x,frow)
%frow indicates which sample (1 or 2) this is testing for
f = [6.67 17.33 42.67 37.33 30.10 29.31 28.74; 6.67 16.11 18.89 15.00 10.56 9.44 8.89];
y = []; % stores the y values at which the spline is evaluated at
for i = x
    for j = 1:size(splineMat,1)
        if (i >= splineMat(j,4)) && (i <= splineMat(j,5))
            first_nested = splineMat(j,2) + (i-splineMat(j,4))*splineMat(j,1); % B + (x-ti)*A
            second_nested = splineMat(j,3) + (i-splineMat(j,4))*first_nested; % C + (x-ti)[B + (x-ti)*A]
            tempy = f(frow,j) + (i-splineMat(j,4))*second_nested;
            y = [y tempy];
        end
    end
end
Find the cubic splines for y1 and y2

```matlab
day = [0 6 10 13 17 20 28]; \% n = 6
f1 = [6.67 17.33 42.67 37.33 30.10 29.31 28.74];
f2 = [6.67 16.11 18.89 15.00 10.56 9.44 8.89];
[cp1,u1,v1,h,z1] = cubic_spline(f1);
[cp2,u2,v2,h,z2] = cubic_spline(f2);
```

Plot the Cubic Splines

```matlab
t = linspace(0,28,100000);
y1 = eval_cubic_spline(cp1,t,1);
y2 = eval_cubic_spline(cp2,t,2);
```

Find the Maximum of Each Spline

```matlab
max1 = max(y1)
max2 = max(y2)
```

Plot the Splines

```matlab
figure
hold on
plot(day,f1,'*k')
plot(t,y1,'--k');
xlabel("x")
ylabel("f(x)")
title("Cubic splines")
legend("Sample 1","CP1")
hold off
figure
hold on
plot(day,f2,'*b')
plot(t,y2,'--b');
xlabel("x")
ylabel("f(x)")
title("Cubic splines")
legend("Sample 2","CP2")
hold off
```