Optional Exam 2 Solutions

Kathryn Atherton

ABE 30100

03/27/2019

Part A

```
t = 0:200:3400; % s
Temp = [25, 196, 396, 511, 551, 551, 538, 533, 529, 520, 516, 516, 511, 511, 507, 507, ...
    502, 498]; % C

S = cubic_spline(t,Temp);
```

`Valid from x = 0.0000 to 200.0000`

f = $1.577\ 10^{-6}\ x^3 + 1.2663\ 10^{-18}\ x^2 + 0.79192\ x + 25.0$

`Valid from x = 200.0000 to 400.0000`

f = $0.98116\ x + 0.0009462\ (x - 200.0)^2 - 4.26\ 10^{-6}\ (x - 200.0)^3 - 0.23188$

`Valid from x = 400.0000 to 600.0000`

f = $0.84844\ x - 0.0016098\ (x - 400.0)^2 + 1.2129\ 10^{-6}\ (x - 400.0)^3 + 56.623$

`Valid from x = 600.0000 to 800.0000`

f = $0.35007\ x - 0.00088207\ (x - 600.0)^2 + 6.5852\ 10^{-7}\ (x - 600.0)^3 + 300.96$

`Valid from x = 800.0000 to 1000.0000`

f = $0.076268\ x - 0.00048695\ (x - 800.0)^2 + 5.2806\ 10^{-7}\ (x - 800.0)^3 + 489.99$

`Valid from x = 1000.0000 to 1200.0000`

f = $6.0423\ 10^{-7}\ (x - 1000.0)^3 - 0.00017012\ (x - 1000.0)^2 - 0.055146\ x + 606.15$

`Valid from x = 1200.0000 to 1400.0000`

f = $0.00019242\ (x - 1200.0)^2 - 0.050685\ x - 3.1998\ 10^{-7}\ (x - 1200.0)^3 + 598.82$

`Valid from x = 1400.0000 to 1600.0000`

f = $4.3136\ 10^{-7}\ (x - 1400.0)^2 - 0.012114\ x - 1.993\ 10^{-7}\ (x - 1400.0)^3 + 549.96$

`Valid from x = 1600.0000 to 1800.0000`

f = $3.6717\ 10^{-7}\ (x - 1600.0)^3 - 0.00011915\ (x - 1600.0)^2 - 0.035857\ x + 586.37$

`Valid from x = 1800.0000 to 2000.0000`

f = $0.00010116\ (x - 1800.0)^2 - 0.039456\ x - 1.9387\ 10^{-8}\ (x - 1800.0)^3 + 591.02$

`Valid from x = 2000.0000 to 2200.0000`

f = $0.000089524\ (x - 2000.0)^2 - 0.0013198\ x - 4.1462\ 10^{-7}\ (x - 2000.0)^3 + 518.64$

`Valid from x = 2200.0000 to 2400.0000`

f = $5.5288\ 10^{-7}\ (x - 2200.0)^3 - 0.00015925\ (x - 2200.0)^2 - 0.015265\ x + 549.58$

`Valid from x = 2400.0000 to 2600.0000`

f = $0.00017248\ (x - 2400.0)^2 - 0.01262\ x - 5.469\ 10^{-7}\ (x - 2400.0)^3 + 541.29$

`Valid from x = 2600.0000 to 2800.0000`

f = $5.0974\ 10^{-7}\ (x - 2600.0)^3 - 0.00015566\ (x - 2600.0)^2 - 0.0092566\ x + 535.07$

`Valid from x = 2800.0000 to 3000.0000`

f = $0.00015018\ (x - 2800.0)^2 - 0.010354\ x - 4.9204\ 10^{-7}\ (x - 2800.0)^3 + 535.99$

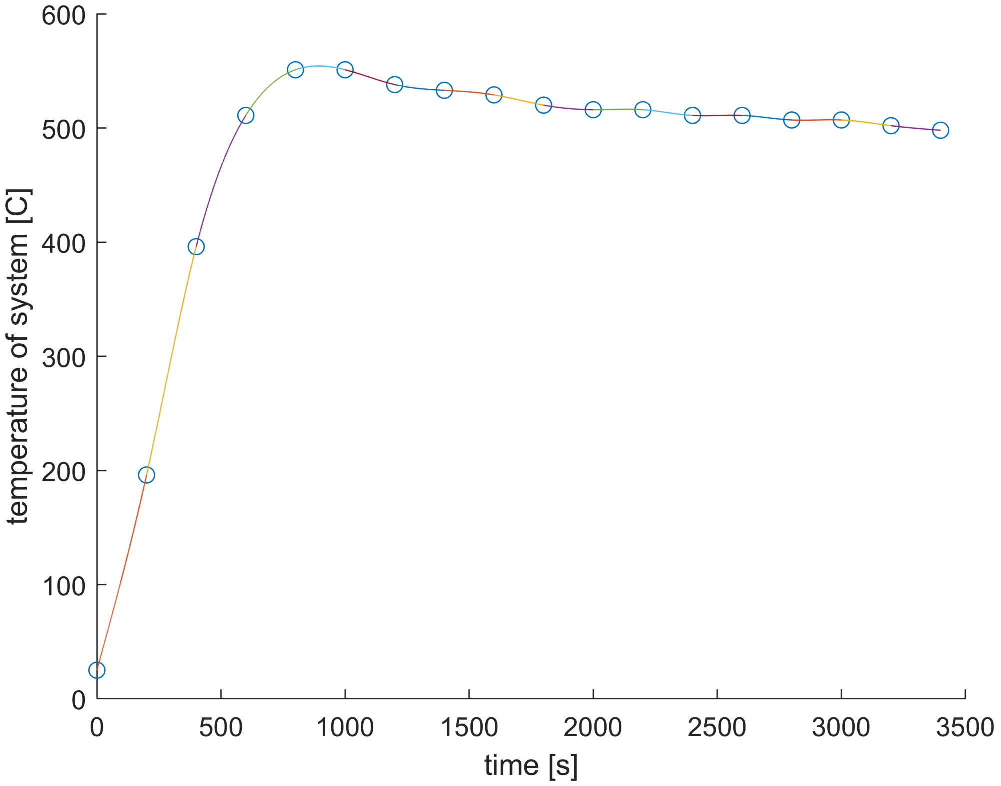
```
Valid from x = 3000.0000 to 3200.0000
```

f = $3.3343\ 10^{-7}\ (x-3000.0)^3 - 0.00014505\ (x-3000.0)^2 - 0.0093278\ x + 534.98$

```
Valid from x = 3200.0000 to 3400.0000
```

f = $0.000055012\ (x-3200.0)^2 - 0.027335\ x - 9.1686\ 10^{-8}\ (x-3200.0)^3 + 589.47$

```
xlabel('time [s]')
ylabel('temperature of system [C]')
```



Part B

```
syms time
% maximum is between 800 and 1000 s
T_v_t_max = 0.076268 * time - 0.00048695 * (time - 800) ^ 2 + 5.2806e-7 * (time - 800) ^ 3 ...
    + 489.99;

max_time = newton_raphson(diff(T_v_t_max), 900, 0.001)
```

```
max_time = 891.9639
```

Answer: At t = 891.96 s, the temperature of the system reaches its maximum.

Part C

```
k = 0.58; % W/m.K
```

```
A = 1; % m^2
d = 0.1; % m
T1 = 25; % C
delta_t = 200; % s
mass = 1; % kg
cp = 1.67 * 1000; % J/kg.K
t = [t(1:5), max_time]; % s
Temp = [Temp(1:5), double(subs(T_v_t_max, time, max_time))]; % C

for i = 1:length(t)
   dQ_dt = k * A * (Temp(i) - T1) / d; % J/s
   Q = dQ_dt * delta_t; % J
   T1 = T1 + Q / (mass * cp); % C
end

double(T1) % C
```

```
ans = 544.0995
```

Answer: The final temperature of the product is 544.10 degrees C.

```
function S = cubic_spline(x1, y)
m = length(x1);                                                % finds the length of the x vect
n = length(y);                                                 % finds the length of the y vect

if m ~= n                                                      % checks that x and y are the sa
                                                                   % length

    error('Error: x and y have different dimensions.');
elseif m < 3                                                   % checks that there are more tha
                                                                   % points

    error('Error: not enough points to create a cubic spline.');
else
    scatter(x1,y);                                             % plots the data as a scatter pl
    hold on;
    [A, B, C, D] = spline_coeff(x1,y);                         % calls the spline coefficient-
                                                                   % function

    S = zeros(m, 1);                                           % makes a vector of zeros to hol
                                                                   % spline functions

    for i = 1:m-1                                              % iterates through the spline
                                                                   % functions (one less than
                                                                   % number of points)

        a = double(A(i));                                      % assigns a variable to the A co
        b = double(B(i));                                      % assigns a variable to the B co
        c = double(C(i));                                      % assigns a variable to the C co
        d = double(D(i));                                      % assigns a variable to the D co
        digits(5)                                              % assigns the number of signific
                                                                   % displayed when the functio
                                                                   % printed

        syms x                                                 % assigns x to be a symbolic var
        fprintf('Valid from x = %.4f to %.4f', x1(i), x1(i + 1))
                                                               % prints the calculated spline
```

```
                                                                   % its valid range
        f = vpa(a) + (vpa(b) * (x - vpa(x1(i)))) + (vpa(c) * ((x - vpa(x1(i))) ^ 2)) + ...
            (vpa(d) * ((x - vpa(x1(i))) ^ 3))
        vals = x1(i):0.01:x1(i+1);                             % makes an array of x values in
                                                                   % range
        x = vals;
        plot(vals, subs(f));                                   % plots the calculated spline fu
        hold on;
    end
end
end

function H = h_matrix(x)
n = length(x);                                                 % finds the length of the x vect
H = zeros(n,n);                                                % creates a square vector with
                                                                   % the size of the x vector
H(1,1) = 1;                                                    % assigns the first and last ele
                                                                   % the diagonal to be 1
H(n,n) = 1;
for i = 2:n-1                                                  % iterates through the rest of t
    for j = 1:n                                                % iterates through the columns
        if j == i                                              % finds the element on the diago
                                                                   % assigns the elements on an
                                                                   % and after the appropriate
                                                                   % functions
            H(i, j) = double(2 * ((x(i) - x(i - 1)) + (x(i + 1) - x(i))));
            H(i, j - 1) = double(x(i) - x(i - 1));
            H(i, j + 1) = double(x(i + 1) - x(i));
        end
    end
end
end

function C = k_matrix(x, y)
m = length(x);                                                 % finds the length of the x vect
n = length(y);                                                 % finds the length of the y vect
if m ~= n                                                      % checks that x and y have the s
    error("Error: x and y have different dimensions.");
else
    K = zeros(m,1);                                            % makes a vector of zeros of the
    for i = 2:m-1                                              % iterates through the middle el
                                                                   % the first and last) and ca
                                                                   % the appropriate variables
        h1 = double(x(i + 1) - x(i));
        h0 = double(x(i) - x(i - 1));
        a2 = double(y(i + 1));
        a1 = double(y(i));
        a0 = double(y(i - 1));
                                                               % calculates the K-function
        K(i,1) = double(((3 * (a2 - a1)) / h1) - ((3 * (a1 - a0)) / h0));
    end
    H = h_matrix(x);                                           % calls the H matrix function
    C = H\K;                                                   % obtains the C coefficients by
```

```
                                                                   % the H matrix by the K vect
end
end

function [A, B, C, D] = spline_coeff(x, y)
m = length(x);                                                 % finds the length of x
n = length(y);                                                 % finds the length of y
if m ~= n                                                      % checks that x and y are the sa
    error('Error: x and y have different dimensions.');
else
    B = zeros(length(y)-1,1);                                  % makes an appropriately sized B
                                                                   % vector

    D = B;                                                     % makes an appropriately sized D
                                                                   % vector

    C = k_matrix(x,y);                                         % calls the K matrix function to
                                                                   % coefficients

    A = y;                                                     % assigns the y values to the A
    for i = 1:m-1                                              % iterates through the lengths o
                                                                   % D vectors and finds the B
                                                                   % coefficients from A, C, an
                                                                   % h variable

        h = (x(i + 1) - x(i));
        B(i,1) = double(((A(i + 1) - A(i)) / h) - (((C(i + 1) + 2 * C(i)) * h) / 3));
        D(i,1) = double((C(i + 1) - C(i)) / (3 * h));
    end
end
end

function [x_root, i] = newton_raphson(f, x1, error_tol)
    time      = x1;
    zero   = double(subs(f));                                  % sets zero to the value of the
                                                                   % function at the given x
                                                                   % point

    x_root = x1;                                               % renames input x value
    i      = 0;                                                % sets iteration counter to zero
    while abs(zero) > error_tol                                % checks to see if another
                                                                   % iteration should be
                                                                   % performed

        time     = x_root;
        slope = double(subs(diff(f)));                         % finds the slope of the functio
                                                                   % at the given point

        if slope == 0                                          % checks for a minimum or maximu
            fprintf('Error: stuck at minimum or maximum of function.\n')
            zero   = 0;                                        % breaks the while loop so that
                                                                   % function doesn't go on for

            x_root = 'N/A';
        else
            b      = zero - slope * x_root;                    % finds the b of the function
                                                                   % y = mx + b

            x_root = double(-b / slope);                       % finds the new x where y = 0 fo
                                                                   % the linear function

            time      = x_root;
            zero   = double(subs(f));                          % finds the value of the functio
```

```
                                                                   % at the x found above
            i      = i + 1;                                    % adds iteration to counter
        end
    end
    if x_root == 'N/A'                                         % changes the zero value to N/A
                                                                   % the case that a maximum wa
                                                                   % found after loop break

        zero  = 'N/A';
    end
end
```