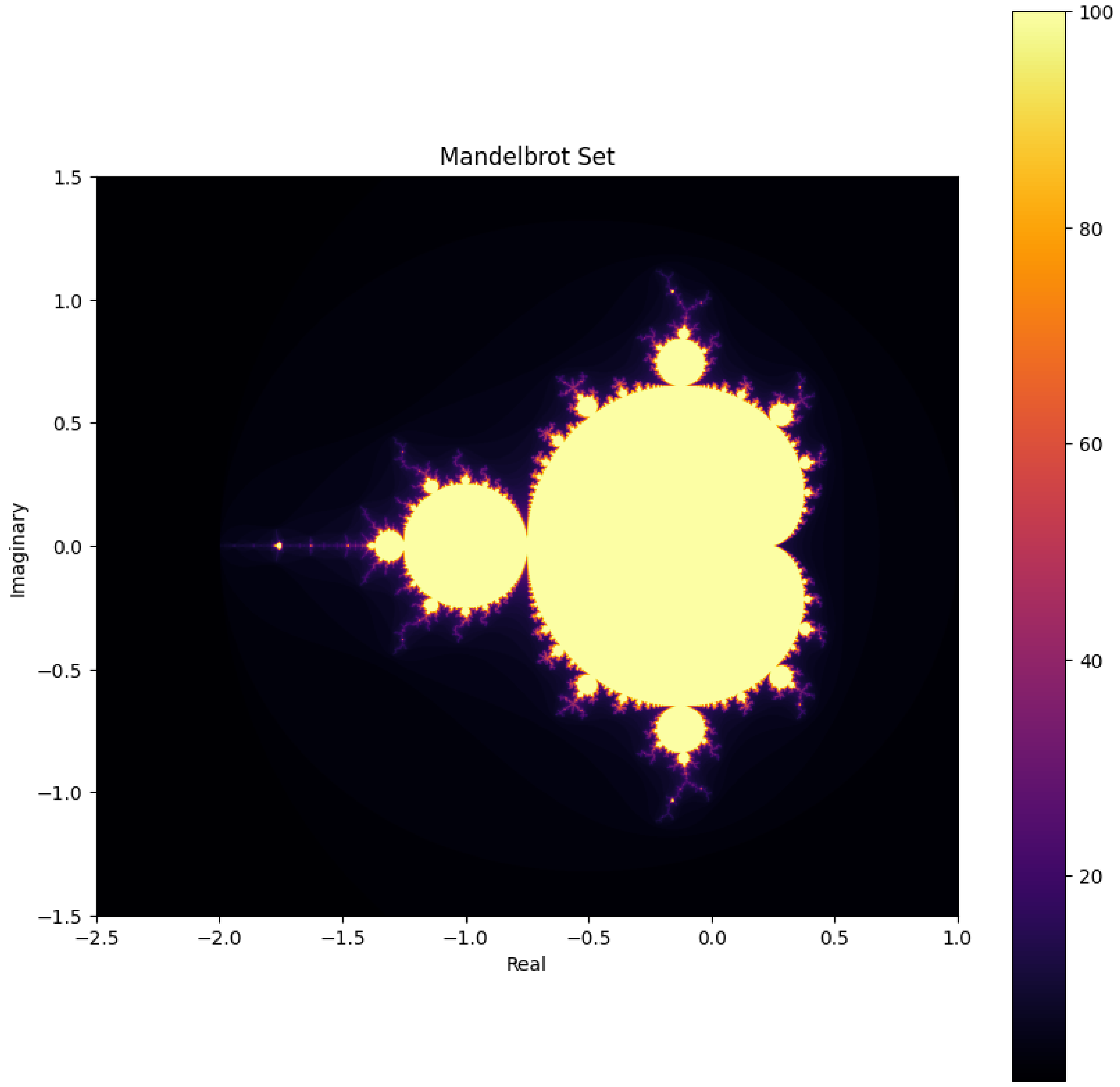
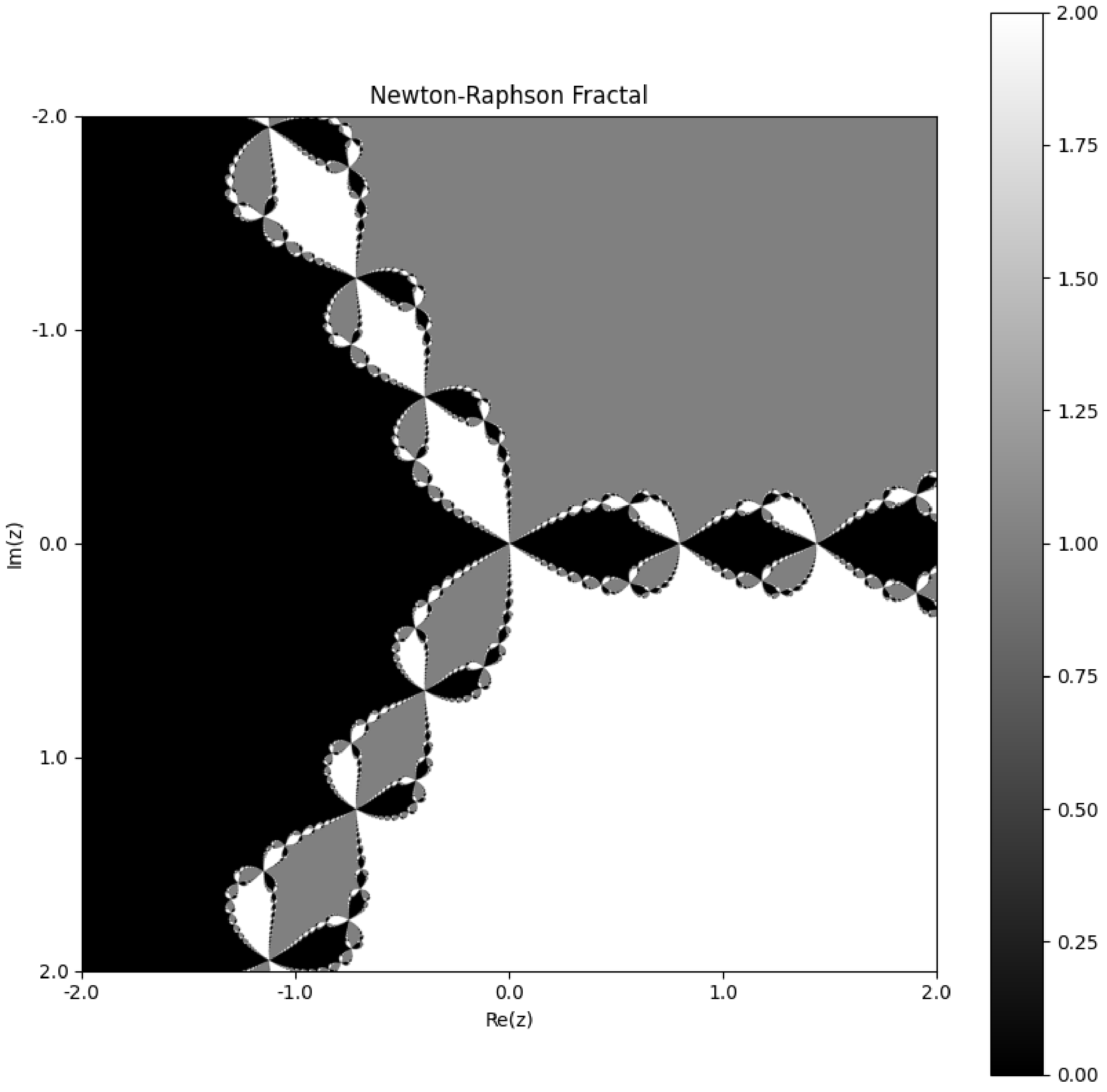
Numerical Methods (DS288): Assignment 3

Name: Subhasis Biswas
Serial Number: 23-1-22571

Question 1:
Question a)



Question b)



Observations and interesting points:

For the Newton's Fractal, one can observe how symmetric the fractal is, and it divides the complex region into three separate regions, divided by 120$^{\o}$. The "bubbles" in each of the three arms has the color of the exact opposite corner (towards which the tip of the bubble is pointing) and the pattern continues seemingly infinitely. Three distinct colors correspond to the three distinct entries: Black for 0, Grey for 1 and White for 2.
For the Mandelbrot's imgplot, the region of convergence is not as simple as one might expect. We can observe the recurrence of the main part of the body all around the boundary of it, although in smaller sizes, as if the pattern keeps on repeating when zoomed in.

Another significant thing I have learned from doing this exercise is the usage of the "decorator" [Definition: `https://wiki.python.org/moin/PythonDecorators`] "@*njit*" using

the "*numba*" module in python.
Without using the "@njit", the program written for the Mandelbrot image plot takes approximately 2 min 30 sec to execute, whereas using the code as suggested in the footnote of the assignment, we halve the execution time, approximately 1 minute. In order to make this even faster, we have used a function named "mandel_img" that performs the nested loop operation

*z=np.asarray([mandelbrot(m+1j*n) for n in y for m in x])*

and applied the decorator on it, reducing the execution time to $\approx$ 7 seconds (21$\times$ speedup).

In the Newton's Fractal code, the speedup is hugely prominent. There are three instances of decorator usage:
1. Computation of the function value at z.
2. Computation of the root by Newton Raphson.
3. Performing *image=np.asarray([NRFrac(m+1j*n) for n in y for m in x])*

Without @njit, time taken was around 25 minutes, and with @njit applied onto all three of them, it took just around 25 seconds (60$\times$ speedup).

Question 2:

a)

We have $p_n(x) = \sum_{i=0}^{n} {}^n l_i(x) f(x_i)$ where ${}^n l_i(x) = \dfrac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$

Before proving the statement for arbitrary n, we can see that

$$\begin{aligned}
&{}^n l_i(x) -{}^{n-1} l_i(x)\\
&= \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} - \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_{n-1})}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_{n-1})}\\
&= \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_{n-1})}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_{n-1})}\left[\frac{(x-x_n)}{(x_i-x_n)}-1\right]\\
&= \frac{(x-x_0)...(x-x_{n-1})}{(x_i-x_0)...(x_i-x_{i-1})...(x_i-x_{i+1})...(x_i-x_n)} = c_i(x-x_0)...(x-x_{n-1})
\end{aligned}$$

with $c_i = \dfrac{1}{(x_i-x_0)...(x_i-x_{i-1})...(x_i-x_{i+1})...(x_i-x_n)} = \dfrac{1}{\prod\limits_{j=0,\ j\neq i}^{n}(x_i-x_j)}$ being a constant. ${}^k l_i(x)$ denotes the Lagrange's ith Fundamental polynomial for degree k (Denoted as $L_{k,i}(x)$ in the book). The equality established above holds for $i = 0$ to $i = n-1$.

$$\begin{aligned}
&\text{Therefore, } p_n(x) - p_{n-1}(x) = \sum_{i=0}^{n-1}[{}^n l_i(x) -{}^{n-1} l_i(x)]f(x_i) +{}^n l_n(x) f(x_n)\\
&= (x-x_0)...(x-x_{n-1})\left[c_0 f(x_0) + ... + c_{n-1}f(x_{n-1}) + \frac{f(x_n)}{(x_n-x_0)...(x_n-x_{n-1})}\right]\\
&= (x-x_0)...(x-x_{n-1})\Big[c_0 f(x_0) + ... + c_{n-1}f(x_{n-1}) + c_n f(x_n)\Big]\\
&= c(x-x_0)...(x-x_{n-1})
\end{aligned}$$

$\square$

b)

Under a permutation of the set $(x_0, ..., x_n)$, the interpolating polynomial $p_n(x)$ remains unchanged due to the following: Suppose $\exists\ q_n(x)$ such that $q_n(x_{\sigma(i})) = f(x_{\sigma(i)})$ for all i, then $\phi_n(x) = q_n(x) - p_n(x)$ is a polynomial of degree at most n with $n+1$ roots $\big[$since σ is a permutation on a finite set, thus bijective, therefore for any $k \in \{0, 1, ...n\}$, $\exists\ i$ such that $\sigma(i) = k \implies q_n(x_{\sigma(i)}) = f(x_{\sigma(i)}) \implies q_n(x_k) = f(x_k)$ & $p_n(x_k) = f(x_k)$ is a given.$\big]$,
which is an impossibility by the fundamental theorem of algebra; thus $\phi_n(x)$ must be identically zero on $J \implies p_n(x) = q_n(x)\ \forall x \in J$.

Thus, in the expression $p_n(x) - p_{n-1}(x)$, the only change dependent on the permutation is $p_{n-1}(x)$, and denote the dependence by ${}^\sigma p_{n-1}(x)$.
Assume, for a general permutation σ acting on $(0, 1, ..., n)$ $[(\cdot,\cdot,\cdot)$ denotes an ordered set$]$, the first n elements $(\sigma(0), ..., \sigma(n-1))$ has the element "k" $(= \sigma(n))$ missing. So, ${}^\sigma p_{n-1}(x)$ interpolates at the nodes $\{x_0, ..., x_{k-1}, x_{k+1}, ..., x_n\}$, i.e ${}^\sigma p_{n-1}(x) = \sum_{i=0}^{k-1} {}^{n-1,\sigma} l_i(x) f(x_i) +$

$$\sum_{i=k+1}^{n} {}^{n-1,\sigma}l_i(x)f(x_i).$$

Now, ${}^{n,\sigma}l_i(x) -^{n-1,\sigma} l_i(x) = {}^{n}l_i(x) -^{n-1,\sigma} l_i(x)$ [since $\{x_0, ..., x_n\} = \{x_{\sigma(0)}, ..., x_{\sigma(n)}\}$, and ${}^{n}l_i$ uses all the points in the set, so ${}^{n,\sigma}l_i = {}^{n}l_i$].

We have,

$${}^{n}l_i(x) -^{n-1,\sigma} l_i(x) = \frac{\displaystyle\prod_{j=0,\ j\neq i}^{n} (x - x_j)}{\displaystyle\prod_{j=0,\ j\neq i}^{n} (x_i - x_j)} - \frac{\displaystyle\prod_{j=0,\ j\neq i, j\neq k}^{n} (x - x_j)}{\displaystyle\prod_{j=0,\ j\neq i, j\neq k}^{n} (x_i - x_j)}$$

$$= \left[\prod_{j=0,\ j\neq i, j\neq k}^{n} (x - x_j)\right]\left[\frac{(x - x_k)}{\displaystyle\prod_{j=0,\ j\neq i}^{n} (x_i - x_j)} - \frac{1}{\displaystyle\prod_{j=0,\ j\neq i, j\neq k}^{n} (x_i - x_j)}\right]$$

$$= \left[\prod_{j=0,\ j\neq i, j\neq k}^{n} (x - x_j)\right]\left[\frac{(x - x_k) - (x_i - x_k)}{\displaystyle\prod_{j=0,\ j\neq i}^{n} (x_i - x_j)}\right]$$

$$= \left[\prod_{j=0,\ j\neq k}^{n} (x - x_j)\right]\left[\frac{1}{\displaystyle\prod_{j=0,\ j\neq i}^{n} (x_i - x_j)}\right] = \left[\prod_{j=0,\ j\neq k}^{n} (x - x_j)\right] c_i,$$ with c_i being the exact same as in problem a)

Thus, $p_n(x) -^{\sigma} p_{n-1}(x) = \displaystyle\sum_{i=0}^{k-1} [{}^{n}l_i(x) - {}^{n-1,\sigma}l_i(x)] + \sum_{i=k+1}^{n} [{}^{n}l_i(x) - {}^{n-1,\sigma}l_i(x)] + {}^{n}l_k(x)$

$$= \left[\prod_{j=0,\ j\neq k}^{n} (x - x_j)\right]\left[c_0 f(x_0) + ... + c_{k-1} f(x_{k-1}) + c_{k+1} f(x_{k+1}) + ... + c_n f(x_n) + \frac{f(x_k)}{\displaystyle\prod_{j=0,\ j\neq k}^{n} (x_i - x_j)}\right]$$

$$= \left[\prod_{j=0,\ j\neq k}^{n} (x - x_j)\right]\left[\sum_{i=0}^{n} c_i f(x_i)\right] = c \prod_{j=0,\ j\neq k}^{n} (x - x_j).$$

Therefore, we finally arrive at $\dfrac{p_n(x) - {}^{\sigma}p_{n-1}(x)}{\displaystyle\prod_{i=0}^{n-1} (x - x_{\sigma(i)})} = c$ for any permutation.

Thus, $f[x_0, ..., x_n] = f[x_{\sigma(0)}, ..., x_{\sigma(n)}]$ holds for any σ.

□

c)
We can write the right hand side as:
$RHS = f[x_0]+[p_1(x)-p_0(x)]+[p_2(x)-p_1(x)]+...+[p_n(x)-p_{n-1}(x)] = p_n(x)-p_0(x)+f[x_0]$.
Since, $p_0(x)$ is the zero-th degree interpolating polynomial satisfying $p_0(x_0) = f(x_0) = f[x_0]$, thus $p_0(x) = f[x_0]$ for all x, which implies $RHS = p_n(x) = LHS$. $\square$.

Before proving the statement regarding the error term, we first prove a lemma:

<u>Lemma</u>: $f[x_0, ..., x_n] = \dfrac{f[x_1, ..., x_n] - f[x_0, ..., x_{n-1}]}{x_n - x_0}$ for any n and any $x_0, ..., x_n$

Proof: We have $f[x_1, ..., x_n] = \sum_{i=1}^{n} c_i^* f(x_i)$ and $f[x_0, ..., x_{n-1}] = \sum_{i=0}^{n-1} c_i^{\#} f(x_i)$

with $c_i^* = \dfrac{1}{\prod_{j=1,\ j\neq i}^{n} (x_i - x_j)}$ and $c_i^{\#} = \dfrac{1}{\prod_{j=0,\ j\neq i}^{n-1} (x_i - x_j)}$. We obtain for $1 \leq i \leq n-1$,

$$c_i^* - c_i^{\#} = \frac{(x_i - x_0) - (x_i - x_n)}{\prod_{j=0,\ j\neq i}^{n} (x_i - x_j)} = (x_n - x_0)c_i.$$

Finally, the from the remaining two terms:

$$c_n^* f(x_n) - c_0^{\#} f(x_0) = \frac{f(x_n)}{\prod_{j=1, j\neq n}^{n} (x_n - x_j)} - \frac{f(x_0)}{\prod_{j=0, j\neq 0}^{n-1} (x_0 - x_j)}$$

$$= \frac{(x_n - x_0) f(x_n)}{\prod_{j=0, j\neq n}^{n} (x_n - x_j)} + \frac{(x_n - x_0) f(x_0)}{\prod_{j=0, j\neq 0}^{n} (x_0 - x_j)} = (x_n - x_0)[c_n f(x_n) + c_0 f(x_0)]$$

Thus, $\dfrac{f[x_1, ..., x_n] - f[x_0, ..., x_{n-1}]}{x_n - x_0} = \dfrac{(x_n - x_0)\sum_{i=0}^{n} c_n f(x_n)}{x_n - x_0} = c = f[x_0, ..., x_n]$ $\square$

Now, we proceed with rest of the argument.

We'll try to prove this by induction on the set $\mathbb{N} \cup \{0\}$ [base case $n = 0$].

[Note: There can be two cases for mathematical induction: One when the base case is $n = 0$ another is $n = 1$ (the usual): `https://en.wikipedia.org/wiki/Mathematical_induction#Description`]

We have for $n = 0$: $f(x) - p_0(x) = \dfrac{f(x) - f(x_0)}{(x - x_0)}(x - x_0) = f[x_0, x](x - x_0)$. So, the statement holds for the base case.
Let us assume that the statement is true for $n = k - 1$. For $n = k$:

$RHS = f[x_0, ..., x_k, x](x - x_0)...(x - x_k) = f[x, x_0, ..., x_{k-1}, x_k](x - x_0)...(x - x_k)$ [using invariance of divided difference under permutation, by problem b)]

Therefore, $RHS = \left[\frac{f[x, x_0, ..., x_{k-1}] - f[x_0, ..., x_k]}{x_k - x}\right](x - x_0)...(x - x_k)$ since the previous lemma holds for any collection of data points.

$= \left[\frac{\frac{f(x) - p_{k-1}(x)}{(x - x_0)...(x - x_{k-1})} - \frac{p_k(x) - p_{k-1}(x)}{(x - x_0)...(x - x_{k-1})}}{x - x_k}\right](x - x_0)...(x - x_k) = f(x) - p_k(x) = LHS$

Thus, our hypothesis holds for all $n \in \mathbb{N} \cup \{0\}$. $\square$

d)

The function $\phi(x) = f(x) - p_n(x)$ has at least $n+1$ distinct zeros at $x_0, ..., x_n$ since $f(x_i) = p_n(x_i)$ for all $1 \le i \le n$. Now, $\phi'(x)$ has at least n zeros, with the zeros being somewhere in the intervals (x_0, x_1), (x_1, x_2),...,(x_{n-1}, x_n), by the Rolle's Theorem. Continuing in this manner, $\phi^{(n)}(x)$ must have at least one zero, lying in between the two zeros of $\phi^{(n-1)}(x)$. Let the zero of $\phi^{(n)}(x)$ be $\xi \in J$.

So, $\phi^{(n)}(\xi) = f^{(n)}(\xi) - p_n^{(n)}(\xi) = f^{(n)}(\xi) - n! f[x_0, ..., x_n] = 0$

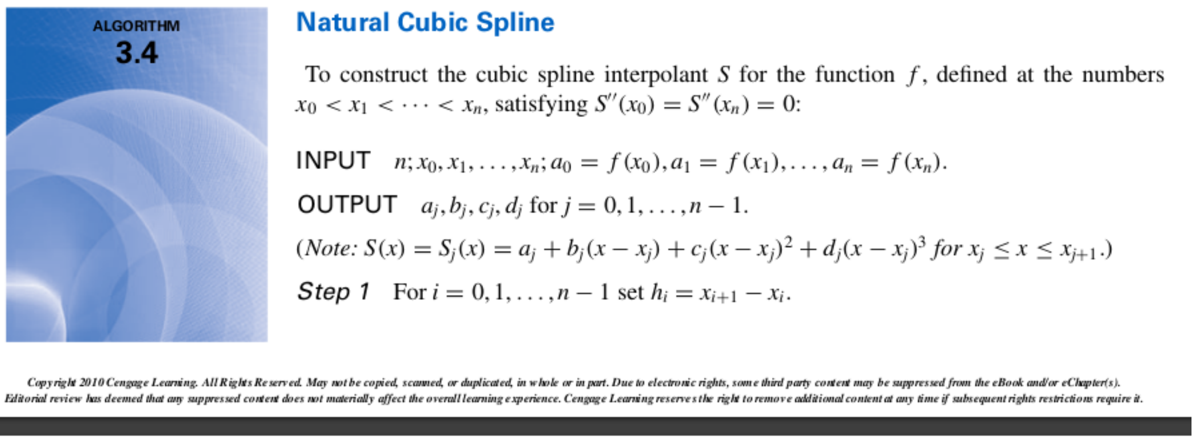
$\implies \frac{f^{(n)}(\xi)}{n!} = f[x_0, ..., x_n]$

Note: $p_n^{(n)}(x) = n! f[x_0, ..., x_n]$ is by the first equality of problem c). The last term involves a monic polynomial of degree n (i.e coefficient of x^n is 1) and all the other terms has degree $< n$. Thus for the $n-th$ order derivative of $p_n(x)$, all those terms vanish and only the last term survives with the polynomial derivative as $n!$ (since monic), along with the constant term $f[x_0, ..., x_n]$ being multiplied.

$\square$

Question 3:

A screenshot of the algorithm 3.4 as given in the book:

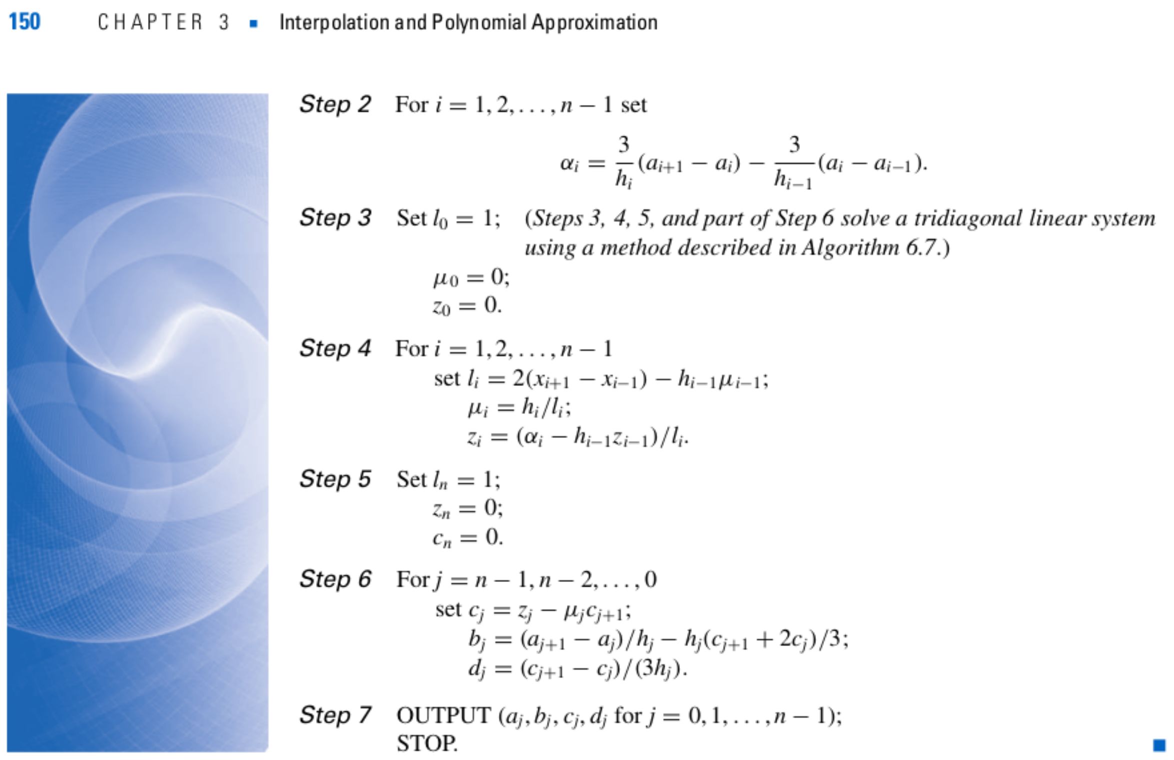


ALGORITHM **3.4**

Natural Cubic Spline

To construct the cubic spline interpolant S for the function f, defined at the numbers $x_0 < x_1 < \cdots < x_n$, satisfying $S''(x_0) = S''(x_n) = 0$:

```
INPUT  $n; x_0, x_1, \ldots, x_n; a_0 = f(x_0), a_1 = f(x_1), \ldots, a_n = f(x_n)$.

OUTPUT  $a_j, b_j, c_j, d_j$ for $j = 0, 1, \ldots, n-1$.

(Note: $S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ for $x_j \le x \le x_{j+1}$.)

Step 1  For $i = 0, 1, \ldots, n-1$ set $h_i = x_{i+1} - x_i$.
```

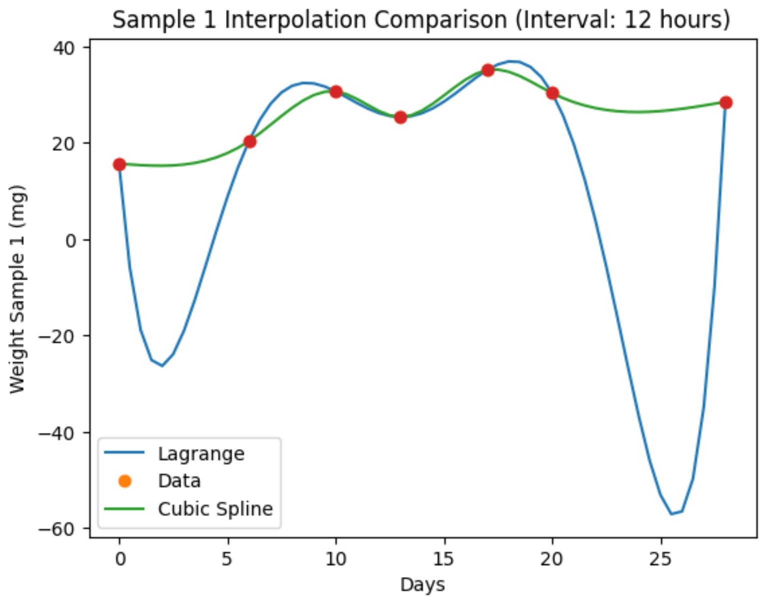


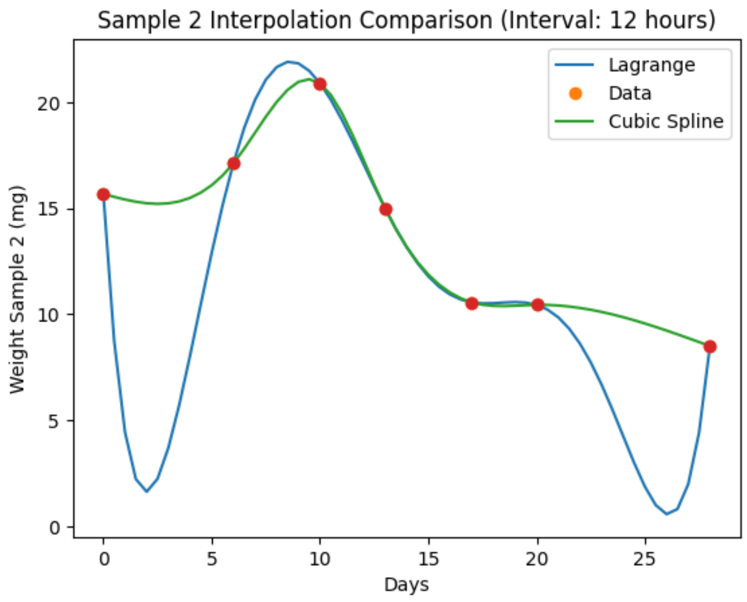
```
Step 2  For $i = 1, 2, \ldots, n-1$ set
            $\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$.

Step 3  Set $l_0 = 1$;  (Steps 3, 4, 5, and part of Step 6 solve a tridiagonal linear system
                         using a method described in Algorithm 6.7.)
            $\mu_0 = 0$;
            $z_0 = 0$.

Step 4  For $i = 1, 2, \ldots, n-1$
            set $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1}$;
                $\mu_i = h_i / l_i$;
                $z_i = (\alpha_i - h_{i-1} z_{i-1}) / l_i$.

Step 5  Set $l_n = 1$;
            $z_n = 0$;
            $c_n = 0$.

Step 6  For $j = n-1, n-2, \ldots, 0$
            set $c_j = z_j - \mu_j c_{j+1}$;
                $b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3$;
                $d_j = (c_{j+1} - c_j)/(3h_j)$.

Step 7  OUTPUT ($a_j, b_j, c_j, d_j$ for $j = 0, 1, \ldots, n-1$);
        STOP.
```

■

Question a) (rounded to 2 decimal places)

	Weight 1				Weight 2			
	a	b	c	d	a	b	c	d
x								
0	15.67	-0.34	0.0	0.03	15.67	-0.28	0.0	0.01
6	20.33	3.01	0.56	-0.17	17.11	1.27	0.26	-0.08
10	30.67	-0.51	-1.44	0.34	20.89	-0.74	-0.76	0.12
13	25.33	0.01	1.62	-0.25	15.0	-2.12	0.3	-0.01
17	35.1	0.84	-1.41	0.2	10.56	-0.3	0.15	-0.02
20	30.31	-2.25	0.38	-0.02	10.44	0.02	-0.05	0.0





Question b)

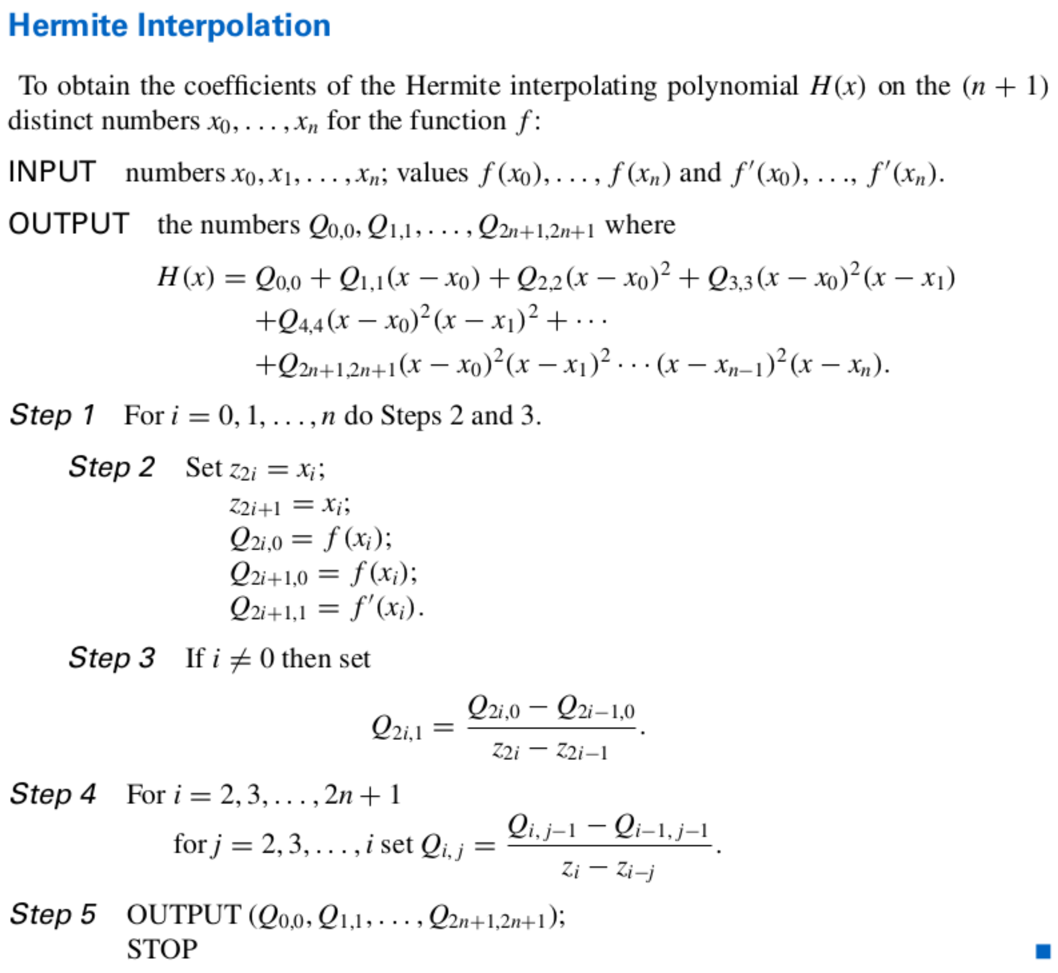
Each piecewise polynomial is cubic, thus in a subinterval the corresponding cubic polynomial can contain at most three roots. So, in that particular subinterval the derivative can change sign at most twice (by Rolle's Theorem), i.e the derivative can have at most two roots.
Using this, we collect all the points x and the midpoints of the subintervals in an ascending order within a list T. One thing to note here is that even though we are bisecting each subinterval, we might get unlucky and miss some sign changes for the derivative if both of the roots of the derivative within that subinterval lie quite close to each other. However, in order to reduce the computation, that's the minimum one can do.
Now, using a function $Df(x)$ that returns the value of the derivative of the spline (using forward divided difference approximation) at a point x, we evaluate the derivatives at all the points of T. If we detect any sign change at two successive points of T, we add the endpoint-pair into a list *root_containing_interval*. Now, for each element in *root_containing_interval*, say (a, b), we apply bisection method on Df using (a, b) as the starting interval with stopping criteria as $|Df(approx)| \leq 10^{-4}$. Append all the roots thus obtained into *roots*.
Thereafter, for all the elements in *roots*, check the sign of the double derivative at that particular root. We can evaluate this by applying the forward divided difference approximation upon Df. If the sign comes out as $-ve$, append the point into a new list called *maxima*.
Now, following the algorithm below, we can find out the global maximum amongst all the local maxima obtained beforehand:

1. Set $globalmax = 0$, $k = 0$.
2. If $spline(roots[k]) > spline(globalmax)$, set $globalmax = roots[k]$ and go to step 3. Otherwise go directly to step 3.
3. $k = k + 1$.
Repeat until all the elements of *roots* are exhausted.

Results:
For Weight Sample 1, Max Average Weight: 35.23, on Day: 17.32
For Weight Sample 2, Max Average Weight: 21.08, on Day: 9.46

Question 4:

Screenshot of the algorithm 3.3 as given in the book:



Hermite Interpolation

To obtain the coefficients of the Hermite interpolating polynomial $H(x)$ on the $(n+1)$ distinct numbers $x_0, \ldots, x_n$ for the function f:

INPUT numbers $x_0, x_1, \ldots, x_n$; values $f(x_0), \ldots, f(x_n)$ and $f'(x_0), \ldots, f'(x_n)$.

OUTPUT the numbers $Q_{0,0}, Q_{1,1}, \ldots, Q_{2n+1,2n+1}$ where

$$\begin{aligned}H(x) = {} & Q_{0,0} + Q_{1,1}(x-x_0) + Q_{2,2}(x-x_0)^2 + Q_{3,3}(x-x_0)^2(x-x_1) \\ & + Q_{4,4}(x-x_0)^2(x-x_1)^2 + \cdots \\ & + Q_{2n+1,2n+1}(x-x_0)^2(x-x_1)^2 \cdots (x-x_{n-1})^2(x-x_n).\end{aligned}$$

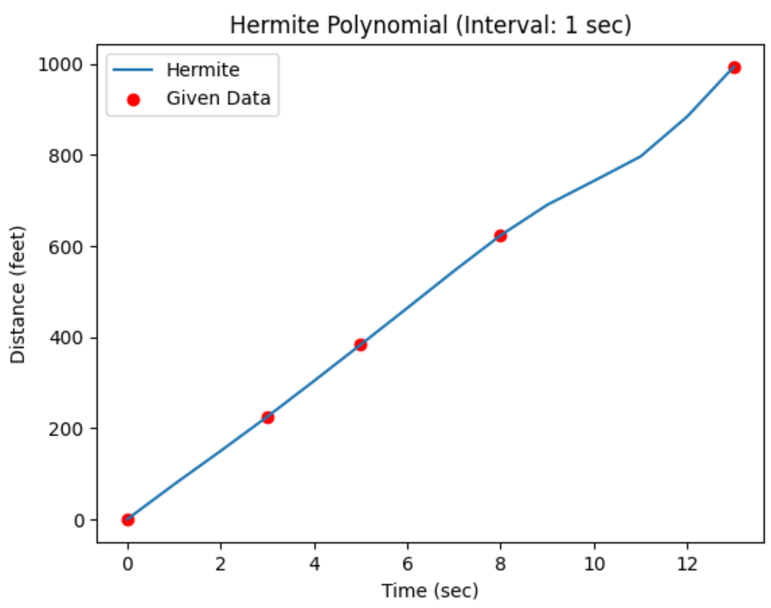
```
Step 1  For $i = 0, 1, \ldots, n$ do Steps 2 and 3.
    Step 2  Set $z_{2i} = x_i$;
                $z_{2i+1} = x_i$;
                $Q_{2i,0} = f(x_i)$;
                $Q_{2i+1,0} = f(x_i)$;
                $Q_{2i+1,1} = f'(x_i)$.
    Step 3  If $i \neq 0$ then set
                $Q_{2i,1} = \frac{Q_{2i,0} - Q_{2i-1,0}}{z_{2i} - z_{2i-1}}$.
Step 4  For $i = 2, 3, \ldots, 2n+1$
            for $j = 2, 3, \ldots, i$ set $Q_{i,j} = \frac{Q_{i,j-1} - Q_{i-1,j-1}}{z_i - z_{i-j}}$.
Step 5  OUTPUT $(Q_{0,0}, Q_{1,1}, \ldots, Q_{2n+1,2n+1})$;
        STOP
```

Here, the variable x is taken as the time, $f(x)$ as the position (distance from the initial point) and $f'(x)$ as the speed of the car. Note that the variable t in the question is being treated as x here.

Question a):

Polynomial Obtained:

$H(x) = -2.0224 \times 10^{-5}x^9 + 0.0010406x^8 - 0.021876x^7 + 0.24304x^6 - 1.5383x^5 + 5.5081x^4 - 10.095x^3 + 7.1619x^2 + 75.0x$ (rounded off to 5 decimal places)



Approximation of Speed at $x = 10$:

Using the forward divided difference approximation formula for derivatives: $f'(x) \approx \dfrac{f(x+h) - f(x)}{(x+h) - x}$, with $h = 10^{-4}$, we obtain $f'(10) = 48.381$

Thus, **Approximate Speed at 10s = 48.381 ft/sec**.

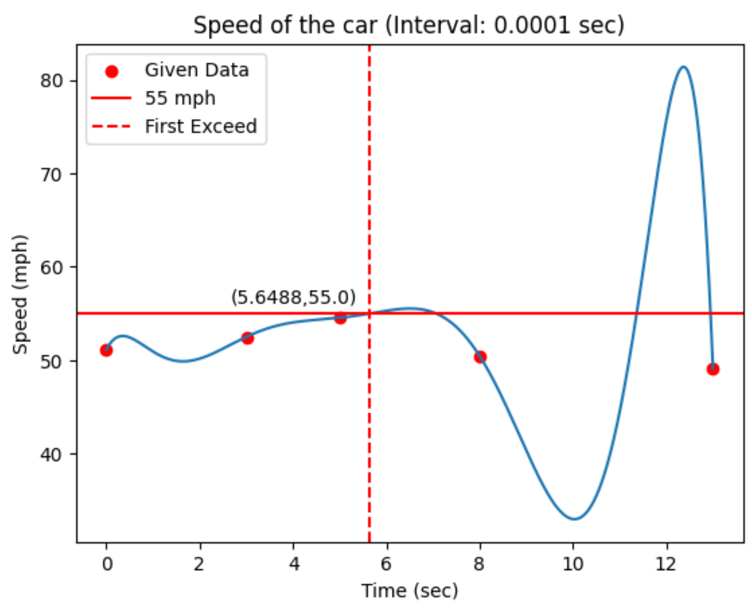
Question b):

To find the speed on the entire interval $[0, 13]$, we have taken the points with spacing 10^{-4} as a list T and evaluated $H'(x)$ on those points using the forward divided difference formula with stepsize $h = 10^{-4}$. Let us call the list of those values v_{fts}, where the speeds are in ft/s. We convert the speed into mph by $v_{mph} = v_{fts} \times 3600/5280$.

We can now find the first instance where the car exceeds $55mph$ by the following:

```
1. Set $k = 0$.
2. If the $k$th element of $v_{mph} \leq 55$, go to step 3, else go to step 4.
3. Set $k = k + 1$ and go to step 2.
4. Print the $k$th element of $T$ and stop iteration.
```

Result: **The car first exceeds 55mph at time 5.6488 sec**



Question c):

We can compute the maximum speed by the following algorithm, using the previous lists T and v_{mph}. Denote the kth element of a list L by $L[k]$.

1. Set $k = 0$, $maxspeed = 0$.
2. If $v_{mph}[k] > maxspeed$, set $maxspeed = v_{mph}[k]$, set $maxtime = T[k]$; move to next step. Else, go directly to the next step.
3. Set $k = k + 1$, go to step 2.

Repeat until all the elements of v_{mph} has been exhausted.
Now, $(maxtime, maxspeed)$ gives us the predicted maximum speed and the corresponding time as per the derivative of the Hermite Interpolating Polynomial.

Result: **The predicted maximum speed is 119.4173 ft/s (81.4209 mph) at time 12.3718 sec**.