Quiz

1) If $A$ and $B$ are both square matrices, then:
   \[ (AB)^{-1} = ? \]
   \[ (AB)^T = ? \]

2) What is the determinant of the following matrix:
   \[ A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \]
Interpolation
Suppose that information about a function $f(x)$ for specified value of $x$ is given in tabular form, where values of $x$ are usually equally spaced.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
</tr>
</tbody>
</table>

**Interpolation** → finding the value of $f(x)$ at some value of $x=1.4$ in between two tabular values, e.g., between $f(x)=3$ and $f(x)=6$.

**Inverse interpolation** → if a value of $f(x)$ between $f(x)=6$ and $f(x)=11$ is known, inverse interpolation is to find the corresponding value of $x$.

**Extrapolation** → determining the value of $f(x)$ at point $x=5$ (outside the range of tabular values).
Lagrange’s Interpolation

The general form

\[ f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) \]  

(5.1)

where

\[ L_i(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0)(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)} \]

and properties that

\[ L_i(x_i) = 1 \]
\[ L_i(x_j) = 0 \quad \text{for} \quad i \neq j \]
Lagrange’s Interpolation

The remainder term

\[ R(x) = (x - x_0)(x - x_1)\ldots(x - x_n)f^{(n+1)}(\xi)/(n + 1)! \]

where

\[ \xi \in (a, b) \text{ if } x, x_1, x_2, \ldots, x_n \in [a, b] \]
Lagrange’s Interpolation

The case $n=1$: Given two points $(x_0, f(x_0))$ and $(x_1, f(x_1))$, there is exactly one line that passes through these points.

$$f_1(x) = L_0(x)f(x_0) + L_1(x)f(x_1)$$

$$= \frac{(x - x_1)}{(x_0 - x_1)}f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)}f(x_1)$$

and the remainder term

$$R = (x - x_0)(x - x_1)f^{(2)}(\xi)/2!$$
Lagrange’s Interpolation

The case $n=2$ (three point case):

$$f_2(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2)$$

$$= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1)$$

$$+ \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2)$$

and the remainder term

$$R = (x-x_0)(x-x_1)(x-x_2)f^{(3)}(\xi)/3!$$
Example

We are given the following data set

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>-6</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
</tr>
</tbody>
</table>

(1) Use the Lagrange interpolation to get a polynomial out of this data.
(2) Determine the degree (or $n$) of the polynomial.
(3) Calculate the remainder.
MATLAB Implementation

% Data
x = 0:3;
fx = [-5, -6, -1, 16];

plot(x,fx,'ro','MarkerSize',10,'MarkerFaceColor','r');

% Create a vector of densely spaced evaluation points
x1 = 0:0.05:3;

% Apply Lagrange interpolation
p1 = (x1-x(2)).*(x1-x(3)).*(x1-x(4))*5/6;
p2 = -3*(x1-x(1)).*(x1-x(3)).*(x1-x(4));
p3 = 0.5*(x1-x(1)).*(x1-x(2)).*(x1-x(4));
p4 = (x1-x(1)).*(x1-x(2)).*(x1-x(3))*16/6;

p = p1 + p2 + p3 + p4;

hold on;
plot(x1,p,'b','LineWidth',2)
axis([-1 4 -10 20]);
MATLAB Implementation

```matlab
function fx = LagrangeInterp(x,y,xk)

n = length(x); % the degree of interpolation polynomial
nk = length(xk); % the number of x-values, where interpolation is to be found

% Make sure that x and y are row vectors
if size(x,1) > 1,  x = x'; end
if size(y,1) > 1,  y = y'; end
if size(x,1) > 1 || size(y,1) > 1 || size(x,2) ~= size(y,2)
    error('both inputs must be equal-length vectors')
end

L = ones(n,nk);

for i = 0:n-1
    for j = 0:(i-1)
        L(j+1,:) = L(j+1,:).*((xk - x(i+1))/(x(j+1)-x(i+1)));
    end
    for j = i+1:n-1
        L(j+1,:) = L(j+1,:).*((xk - x(i+1))/(x(j+1)-x(i+1)));
    end
end

fx = y * L;

plot(x,y,'bo',xk,fx,'r');
```
MATLAB Implementation
Newton’s Interpolation

Linear interpolation is the simplest form of interpolation to connect two data points with a straight line.

Using points two green points \((x_0, f(x_0))\) and \((x_1, f(x_1))\) the interpolant is given by

\[
\frac{f_L(x) - f(x_0)}{x - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}
\]

or after rearranging:

\[
f_L(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) \quad (5.2)
\]
Newton’s Interpolation

\[
\frac{f(x_1) - f(x_0)}{x_1 - x_0}
\]

is the slope of the red line that connects points \((x_0, f(x_0))\) and \((x_1, f(x_1))\), also called 1\textsuperscript{st} order finite-divided-difference approximation.

Quadratic interpolation is the interpolation of a given data set by using curvature (or polynomial of 2\textsuperscript{nd} order). Quadratic interpolation connects three data points \((x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))\). The equation is given by

\[
f_Q(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) \quad (5.3)
\]

At \(x = x_0\) : \(b_0 = f(x_0)\)

At \(x = x_1\) : \(b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}\)
Newton’s Interpolation

The general form of Newton’s interpolating polynomial:

\[ f_n(x) = f(x_0) + (x - x_0)f[x_1, x_0] + (x - x_0)(x - x_1)f[x_2, x_1, x_0] + \ldots \]

\[ + (x - x_0)(x - x_1) \ldots (x - x_{n-1})f[x_n, x_{n-1}, \ldots, x_0] \]

\[ f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j} \]

\[ f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k} \]

\[ f[x_n, x_{n-1}, \ldots, x_1, x_0] = \frac{f[x_n, x_{n-1}, \ldots, x_1] - f[x_{n-1}, x_{n-2}, \ldots, x_0]}{x_n - x_0} \]
Example of Newton’s Interpolation

Estimate the quadratic function $f(x) = x^2$ at $x = 2.5$ using linear interpolation where the available data points are at $x = 1$, $x = 3$, $x = 5$. 
Example of Newton’s Interpolation

Estimate the quadratic function $f(x) = x^2$ at $x = 2.5$ using linear interpolation where the available data points are at $x = 1, x = 3, x = 5$.

First we estimate at $x_0 = 1$ and $x_1 = 5$

Using the linear interpolation formula where $f(x_0) = 1$ and $f(x_1) = 25$ gives

$$f(x) = 1 + \frac{25 - 1}{5 - 1}(2.5 - 1) = 10$$

which represents a relative error of $\epsilon = 60\%$ (red line).
Example of Newton’s Interpolation

Using the smaller interval between $x_0 = 1$ and $x_1 = 3$ yields

$$f(x) = 1 + \frac{9 - 1}{3 - 1} (2.5 - 1) = 7$$

The shorter interval also gives smaller relative error $\epsilon = 12\%$ (grey line).

If we fit a second-order polynomial to the three points used in the example:

$$x_0 = 1 \quad f(x_0) = 1$$
$$x_1 = 3 \quad f(x_1) = 9$$
$$x_2 = 5 \quad f(x_2) = 25$$
Example of Newton’s Interpolation

and from the equations given for quadratic interpolation

\[
b_0 = f(x_0) = 1
\]

\[
b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{9 - 1}{3 - 1} = 4
\]

\[
b_2 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{25 - 9}{5 - 3} - \frac{9 - 1}{3 - 1} = 1
\]

Substituting these values into the quadratic interpolation

\[f_Q(x) = 1 + 4(x - 1) + 1(x - 1)(x - 3)\] which can be evaluated at \(x = 2.5\) for \(f_Q(2.5) = 6.25\), hence \(\epsilon = 0\%\).
Recall the truncation error for the Taylor series in Lecture 1:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n + 1)!} (x_{i+1} - x_i)^{n+1}$$

For an $n$th-order interpolating polynomial, an analogous relationship for the error is given by

$$R_n = \frac{f^{(n+1)}(\xi)}{(n + 1)!} (x - x_0)(x - x_1) \cdots (x - x_n) \quad (5.4)$$

Where $\xi$ is somewhere in the interval containing the unknown and the data.
Errors

Another alternative formula for error uses a finite divided difference

\[ R_n = f[x_{n+1}, x_n, x_{n-1}, \ldots, x_0](x - x_0)(x - x_1) \cdots (x - x_n) \]

(5.5)
Example

Fit a second-order polynomial to the three points below:

\[ x_0 = 1 \quad f(x_0) = 0 \]
\[ x_1 = 4 \quad f(x_1) = 1.386294 \]
\[ x_2 = 6 \quad f(x_2) = 1.791759 \]

and then use equation (5.5) to estimate the error for the polynomial. Use the additional data point \( f(x_3) = f(5) = 1.609438 \) to obtain your results.

Note that the values of \( f(x) \) were derived from a function of natural logarithm or \( \ln(x) \).
Example

Solution:

\[ b_0 = f(x_0) = 0 \]

\[ b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1.386294 - 0}{4 - 1} = 0.4620981 \]

\[ b_2 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} - b_1 = \frac{0.4620981}{x_2 - x_0} = -0.0518731 \]
Example

```matlab
x = [1,4,6];
y = [0,1.386294,1.791759];

b0 = y(1);
b1 = (y(2)-y(1))/(x(2)-x(1));
b2 = (((y(3)-y(2))/(x(3)-x(2))) - b1)/(x(3)-x(1));

x1 = linspace(1,6,200);
y1 = b0 + b1*(x1-x(1)) + b2*(x1-x(1)).*(x1-x(2));

hold on;
plot(x,y,'bo','MarkerSize',10,'MarkerFaceColor','b');
plot(x1,y1,'r');
hold off;
axis([0 7 0 2]);
```
Example
Example

The first divided differences for the problem are

\[ f[x_1, x_0] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = 0.4620981 \]

\[ f[x_2, x_1] = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = 0.2027326 \]

\[ f[x_3, x_2] = \frac{f(x_3) - f(x_2)}{x_3 - x_2} = 0.1823216 \]
Example

The second divided differences for the problem are

\[
f[x_2, x_1, x_0] = \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0} = -0.05187311
\]

\[
f[x_3, x_2, x_1] = \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1} = -0.020411
\]

The third divided difference is

\[
f[x_3, x_2, x_1, x_0] = \frac{f[x_3, x_2, x_1] - f[x_2, x_1, x_0]}{x_3 - x_0} = 0.007865529
\]
Example

The value for $f_2(2) = 0.5658444$, which represents an error from the exact value of $\ln(2)$ of $0.6931472 - 0.56584444 = 0.1273028$.

If we had not known the exact value, as is most usually the case, we use eqn. (5.5) along the additional value at $x_3$ to estimate the error, as in

$$R_2 = f[x_3, x_2, x_1, x_0](x - x_0)(x - x_1)(x - x_2)$$

$$= 0.007865529(x - 1)(x - 4)(x - 6)$$

$$= 0.007865529(2 - 1)(2 - 4)(2 - 6)$$

$$= 0.0629242$$
The problem with a linear function is that, the first derivative is not continuous at the boundary between two adjacent intervals, thus higher order functions are often required for more accurate results.
Spline Interpolation

The general idea of a spline is: on each interval between data points, the graph is represented with a function. In general there are three spline interpolations: linear, quadratic, and cubic splines.

**Linear splines** - The simplest connection between two data points. The first-order splines for a group of ordered data points can be defined as a set of linear functions:

\[
\begin{align*}
  f(x) &= f(x_0) + m_0(x - x_0) & x_0 \leq x \leq x_1 \\
  f(x) &= f(x_1) + m_1(x - x_1) & x_1 \leq x \leq x_2 \\
  &\vdots \\
  f(x) &= f(x_{n-1}) + m_{n-1}(x - x_{n-1}) & x_{n-1} \leq x \leq x_n
\end{align*}
\]
Spline Interpolation

where $m_i$ is the slope of the straight line connecting the points:

$$m_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \quad (5.6)$$

Linear splines are just like the linear Newton’s’s interpolation we looked at earlier.

Cubic splines
The idea of cubic splines is to construct a curve by using a different cubic polynomial curve between each two data points. Or, pieces of different cubic curves are glued together to form a global curve/function.
Cubic splines are the most popular. They produce an interpolated function that is continuous through to the second derivative. It is generally agreed that cubic is the most optimal degree for splines.
Cubic Splines

The objective in cubic splines is to derive a third-order polynomial for each interval between knots:

\[ f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \]  \hspace{1cm} (5.7)

For \( n+1 \) data points \( (i=0,1,2,\ldots,n) \) there are \( n \) intervals, and, consequently, \( 4n \) unknown constants to evaluate.

Because each pair of knots is connected by a cubic, the second derivative within each interval is a straight line. To verify this observation, we differentiate eqn. (5.7) twice, as in
Cubic Splines

\[ f'_i(x) = 3a_i x^2 + 2b_i x + c_i \]

\[ f''_i(x) = 6a_i x + 2b_i \]

The second derivatives can be represented by a first-order Lagrange interpolating polynomial

\[ f''_i(x) = f''_i(x_{i-1}) \frac{x - x_i}{x_{i-1} - x_i} + f''_i(x_i) \frac{x - x_{i-1}}{x_i - x_{i-1}} \]  \hspace{1cm} (5.8)

Where \( f''_i(x) \) is the value of the second derivative at any point \( x \) within the \( i \)th interval.
Cubic Splines

Equation (5.8) is a straight line connecting the second derivative at the first knot \( f''(x_{i-1}) \) with the second derivative at the second knot \( f''(x_i) \).

Equation (5.8) can be integrated twice to yield an expression for \( f_i(x) \). However, this expression will contain two unknown constants of integration. These constants can be evaluated by invoking the function-equality conditions, that is \( f(x) \) must equal \( f(x_{i-1}) \) at \( x_{i-1} \) and \( f(x) \) must equal \( f(x_i) \) at \( x_i \).

Performing these evaluations gives the cubic equation for each interval:
Cubic Splines

\[ f_i(x) = \frac{f''(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{f''(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1})^3 \]

\[ + \left[ \frac{f(x_{i-1})}{x_i - x_{i-1}} - \frac{f''(x_{i-1})(x_i - x_{i-1})}{6} \right] (x_i - x) \]

\[ + \left[ \frac{f(x_i)}{x_i - x_{i-1}} - \frac{f''(x_i)(x_i - x_{i-1})}{6} \right] (x - x_{i-1}) \]

(5.9)

This equation contains only two unknowns, which are the second derivatives at the end of each interval. These unknowns can be evaluated using the following:
Cubic Splines

\[(x_i - x_{i-1})f''(x_{i-1}) + 2(x_{i+1} - x_i)f''(x_i)\]
\[+ (x_{i+1} - x_i)f''(x_{i+1})\]
\[= \frac{6}{x_{i+1} - x_i}(f(x_{i+1}) - f(x_i))\]
\[+ \frac{6}{x_i - x_{i-1}}(f(x_{i-1}) - f(x_i))\]

(5.10)
Another Way to Derive Cubic Splines

We start from a table of points \([x_i, y_i]\) for \(i = 0, 1, \ldots, n\) for the function \(f(x)\). That makes \(n+1\) points and \(n\) intervals between them. Each interval is to be fitted with a cubic polynomial \(C_i\) making the spline interpolant function \(S(x)\):

\[
S(x) = \begin{cases} 
C_1(x), & x_0 \leq x \leq x_1 \\
C_i(x), & x_{i-1} \leq x \leq x_i \\
C_n(x), & x_{n-1} \leq x \leq x_n 
\end{cases}
\tag{5.11}
\]

where each \(C_i\) is a cubic function that takes general form

\[
C_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \quad i = 1, 2, \ldots, n
\]

To determine the cubic spline \(S(x)\) we must determine the coefficients \(a_i, b_i, c_i,\) and \(d_i\) for each \(C_i\).
Another Way to Derive Cubic Splines

Since there are \( n \) intervals, consequently, there are \( 4n \) unknowns coefficients to determine.

We get two conditions for each interval from the requirement that each cubic polynomial matches the values of the table at both ends of the interval:

\[
C_i(x_{i-1}) = y_{i-1} \quad \text{and} \quad C_i(x_i) = y_i
\]

Or, in other words

\[
a_i x_{i-1}^3 + b_i x_{i-1}^2 + c_i x_{i-1} + d_i = y_{i-1}
\]

and

\[
a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i = y_i
\]

These give us \( 2n \) conditions
Another Way to Derive Cubic Splines

Since we would like to make the interpolation \( S(x) \) as smooth as possible, we require that the first and second derivatives of the cubic \( C_i \) also be continuous, thus:

\[
C''_i(x_i) = C''_{i+1}(x_i)
\]

\[
C'''_i(x_i) = C'''_{i+1}(x_i)
\]

at all the internal points, i.e., \( x_1, x_2, \ldots, x_{n-1} \), resulting in \( 2(n-1) \) conditions. In terms of the points, these conditions can be written as:

\[
3a_i x_i^2 + 2b_i x_i + c_i = 3a_{i+1} x_i^2 + 2b_{i+1} x_i + c_{i+1}
\]

\[
6a_i x_i + 2b_i = 6a_{i+1} x_i + 2b_{i+1}
\]
Another Way to Derive Cubic Splines

So far we have 4n-2 equations, thus we are 2 equations short of being able to determine all the coefficients (which all give 4n equations). There are some standard choices left to the user:

1. Natural boundary conditions

\[ C_1''(x_0) = 0 \quad \text{and} \quad C_n''(x_n) = 0 \]

2. Clamped boundary conditions

\[ C_1'(x_0) = f'(x_0) \quad \text{and} \quad C_n'(x_n) = f'(x_n) \]
MATLAB Implementation

An experiment was conducted to track the growth of cancer cells grown in a specific culture for fifty hours. The number of cells was counted every 5 hours. Two vectors representing the number of counting time and the number of cell population (in thousands) are:

```matlab
hours = 0:5:50;
p = [30 47.2 70.2 77.4 36.3 20.6 18.1 21.4 22 25.4 27.1];
```

We want to know cell number estimates at 13, 22, and 42 hours by using linear interpolation:

```matlab
h_est = [13 22 42];
p_est = interp1(hours, p, h_est, 'linear');
```
MATLAB Implementation

\[ p_{\text{est}} = \begin{bmatrix} 74.52 \ 30.02 \ 23.36 \end{bmatrix} \]
MATLAB Implementation

More accurate results are possible by specifying cubic \texttt{spline} interpolation, a method that connects the given data points by 3$^{rd}$ order polynomials over each interval.

Now we interpolate within the data at every hour:

\begin{verbatim}
    h_est = 0:0.1:50; % refining time points to 0.1
    p_est = interp1(hours, p, h_est, 'spline');
\end{verbatim}
MATLAB Implementation
Multidimensional Interpolation

Two dimensional interpolation deals with determining intermediate values for functions of two variables, \( z = f(x_i, y_i) \). Linear 2D interpolation is formulated as

\[
f(x_i, y_i) = \frac{(x_i - x_2)(y_i - y_2)}{(x_1 - x_2)(y_1 - y_2)} f(x_1, y_1) \\
+ \frac{(x_i - x_1)(y_i - y_2)}{(x_2 - x_1)(y_1 - y_2)} f(x_2, y_1) \\
+ \frac{(x_i - x_2)(y_i - y_1)}{(x_1 - x_2)(y_2 - y_1)} f(x_1, y_2) \\
+ \frac{(x_i - x_1)(y_i - y_1)}{(x_2 - x_1)(y_2 - y_1)} f(x_2, y_2)
\]

(5.12)
Example

The following is the data of temperatures at a number of coordinates on the surface of a rectangular heated plate:

\[
\begin{align*}
T(2,1) &= 60 & T(9,1) &= 57.5 \\
T(2,6) &= 55 & T(9,6) &= 70
\end{align*}
\]

Estimate the temperature at \( x_i = 5.25 \) and \( y_i = 4.8 \).
Example

Solution:

Substituting these values into eqn. (5.12) gives

\[
f(5.25, 4.8) = \frac{(5.25 - 9)(4.8 - 6)}{(2 - 9)(1 - 6)} 60 + \frac{(5.25 - 2)(4.8 - 6)}{(9 - 2)(1 - 6)} 57.5
\]

\[
+ \frac{(5.25 - 9)(4.8 - 1)}{(2 - 9)(6 - 1)} 55 + \frac{(5.25 - 2)(4.8 - 1)}{(9 - 2)(6 - 1)} 70
\]

\[= 61.2143\]
Inverse Interpolation

From Newton’s linear interpolation formula

\[
\frac{f(x) - f(x_0)}{x - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}
\]

we rearrange to get linear inverse interpolation given by

\[
\frac{x - x_0}{x_1 - x_0} = \frac{f(x) - f(x_0)}{f(x_1) - f(x_0)}
\]

or

\[
x = x_0 + \frac{f(x) - f(x_0)}{f(x_1) - f(x_0)} (x_1 - x_0)
\] (5.13)
Inverse Interpolation

Consider a table of values derived from \( f(x) = \frac{1}{x} \)

\[
\begin{array}{ccccccc}
  x & 1  & 2  & 3  & 4  & 5  & 6  \\
  f(x) & 1  & 0.5 & 0.3333 & 0.25 & 0.2 & 0.1667 \\
\end{array}
\]

Determine the value of \( x \) that correspond to \( f(x) = 0.4 \)

To solve the problem, we use a simple approach by fitting a quadratic polynomial to the three points \((2, 0.5), (3, 0.333),\) and \((4, 0.25)\).
References