1-2. Use both linearization algorithms that we studied in class (Algorithm 1 that uses partial derivatives and Algorithm 2 which does not) to estimate the range of the function $f(x) = 2x + 2x^2$ on the interval $[0, 0.2]$. Compare the two estimates with the exact range -- which you should compute by using calculus.

**Algorithm 1**

$$\Delta y = \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i} \right| \cdot \Delta x$$

$$\Delta y = \left| f'(0.1) \right| \cdot \Delta x$$

$$\Delta y = \left| (2 + 4(0.1)) \right| \cdot \Delta x = \left| (2.4) \right| \cdot \Delta x$$

$$\frac{df}{dx} \geq 0, \text{ so } \Delta x = \Delta = 0.1$$

$$\Delta y = \left| (2.4) \right| (0.1) = 0.24$$

$$y = [f(m(x)) - \Delta y, f(m(x)) + \Delta y]$$

$$= [0.24 - 0.24, 0.24 + 0.24]$$

$$= [0.22, 0.48]$$

**Algorithm 2**

$$\Delta y = \left| f(x + \Delta x) - f(m(x)) \right|$$

$$= \left| f(0.1 + 0.1) - f(0.1) \right| = \left| f(0.2) - f(0.1) \right|$$

$$= \left| 0.48 - 0.22 \right| = 0.26$$

$$y = [f(m(x)) - \Delta y, f(m(x)) + \Delta y]$$

$$= [0.24 - 0.26, 0.24 + 0.26]$$

$$= [-0.02, 0.46]$$

Better result than Algorithm 1, an enclosure of the actual range.
3. If use the simplest "and"- and "or"-operations min and max, and our degrees of belief in A, B, and C are, correspondingly, 0.5, 0.6, and 0.7, what is our degree of belief in \((A \lor B) \land C\)?

\[
f_v(0.5, 0.6) = \max(0.5, 0.6) = 0.6
\]

\[
(A \lor B) \land C = f_\land(f_v(A, B), C) = f_\land(0.6, 0.7)
\]

\[
f_\land(0.6, 0.7) = \min(0.6, 0.7) = 0.6
\]
4. For a membership function $\mu(x) = 1 - |1 - x|$, what are the $\alpha$-cuts corresponding to $\alpha = 0.6$? to $\alpha = 0.7$?

$\alpha = 0.6$

$\mu(x) \geq \alpha$

$-|1-x| \geq 0.6$

$-|1-x| \geq 0.9$

$|1-x| \leq 0.4$

$-0.4 \leq 1-x \leq 0.4$

$0.9 \geq x-1 \geq -0.4$

$1.9 \geq x \geq 0.6$

$x = [0.6, 1.4]$

$\alpha = 0.7$

$\mu(x) \geq \alpha$

$-|1-x| \geq 0.7$

$-|1-x| \geq 0.3$

$|1-x| \leq 0.3$

$-0.3 \leq 1-x \leq 0.3$

$0.3 \geq x-1 \geq -0.3$

$1.3 \geq x \geq 0.7$

$x = [0.7, 1.3]$
5-6. Let us assume that the quantity $x$ is described by the membership function $\mu(x) = 1 - |x|$, and the quantity $y$ is described by the membership function $\mu(y) = 1 - |1 - y|$. Use the values $\alpha = 0.2, 0.4, 0.6, 0.8, \text{ and } 1.0$ to form membership functions for $z = x - y$ and $t = x \cdot y$.
\[ \alpha = 0.8 \]

\[ \mu(x) \geq \alpha \]
\[ 1 - |x| > 0.8 \]
\[ |x| < 0.2 \]
\[ x(\alpha) = [-0.2, 0.2] \]

\[ \mu(y) \geq \alpha \]
\[ 1 - |1-y| > 0.8 \]
\[ |1-y| < 0.2 \]
\[ 0.2 \leq y-1 \leq 0.2 \]
\[ y(\alpha) = [0.8, 1.2] \]
\[ z(\alpha) = x(\alpha) - y(\alpha) \]
\[ = [-0.2, 0.2] - [0.8, 1.2] \]
\[ = [-1.4, -0.6] \]
\[ t(\alpha) = x(\alpha) + y(\alpha) \]
\[ = [-0.2, 0.2] + [0.8, 1.2] \]
\[ = [-0.24, 0.24] \]

\[ \alpha = 1 \]
\[ \mu(x) \geq \alpha \]
\[ 1 - |x| \geq 1 \]
\[ |x| \leq 0 \]
\[ x(\alpha) = 0 \]

\[ \mu(y) \geq \alpha \]
\[ 1 - |1-y| \geq 1 \]
\[ |1-y| \leq 0 \]
\[ 1-y = 0 \]
\[ y(\alpha) = 1 \]
\[ z(\alpha) = x(\alpha) - y(\alpha) \]
\[ = 0 - 1 \]
\[ = -1 \]
\[ t(\alpha) = x(\alpha) * y(\alpha) \]
\[ = (0) * (1) \]
\[ = 0 \]
7. Follow the first three steps of bisection to locate the square root of 6, i.e., the solution to the equation $x^2 - 5 = 0$ on the interval $[0, 4]$.

1) $f(0) = -5$
   $f(2) = 2^2 - 5 = -1$  \( \Rightarrow \) interval is now $[2, 4]$
   $f(4) = 4^2 - 5 = 16 - 5 = 11$

2) $f(2) = -1$
   $f(3) = 3^2 - 5 = 9 - 5 = 4$  \( \Rightarrow \) interval is now $[2, 3]$
   $f(4) = 11$

3) $f(2) = -1$
   $f(2.5) = (2.5)^2 - 5 = 6.25 - 5 = 1.25$  \( \Rightarrow \) interval is now $[2, 2.5]$
   $f(3) = 4$

   Solution is inside interval $[2, 2.5]$
8. Write down a Java method that, given a function $f$, values $x_1, ..., x_n$, and a number $h$, computes the array of values

$$
c_i = (f(x_1, ..., x_i, x_{i+1}, ..., x_n) + f(x_1, ..., x_{i-1}, x_{i-1} - h, x_{i+1}, ..., x_n) - 2f(x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n))/h^2.
$$
corresponding to $i = 1, ..., n$.

```java
public double[] evaluate(Function f, double[] params, double h) {
    int n = params.length;
    double[] c = new double[n];
    double h_sq = Math.pow(h, 2);
    for (int i = 0; i < n; i++) {
        double term1, term2, term3;
        params[i] += h; // $x_i + h$
        term1 = f.eval(params);
        params[i] -= 2*h; // $x_i - h$
        term2 = f.eval(params);
        params[i] += h; // $x_i$
        term3 = 2 * f.eval(params);
        c[i] = (term1 + term2 - term3) / h_sq;
    }
    return c;
}
```
9. Suppose that we have three alternatives, for which the gains are in the intervals \([0, 5], [-4, -1], \) and \([-3, 10]\). Is there an alternative which is guaranteed to be optimal (i.e., for which the gain is the largest)?

- list all alternatives which can be optimal;
- which alternative(s) should we select if we use Hurwicz optimism-pessimism criterion with \(\alpha = 0, 0.5, \) and \(1.\)

- **Optimality Guarantor**: \(\frac{x_i}{x_j} \geq \max (x_j), i \neq j\)
  - \([0, 5]\): \(0 \geq \max (-1, 10) \rightarrow\) not guaranteed
  - \([-4, -1]\): \(-4 \geq \max (5, 10) \rightarrow\) not guaranteed
  - \([-3, 10]\): \(-3 \geq \max (5, -1) \rightarrow\) not guaranteed

\[\Rightarrow \text{There is no guaranteed optimal strategy}\]

- **Possible Optimality**: \(\frac{x_i}{x_j} \geq \max (x_j), i \neq j\)
  - \([0, 5]\): \(5 \geq \max (-3, -9) \rightarrow\) can be optimal
  - \([-4, -1]\): \(-1 \geq \max (0, -3) \rightarrow\) cannot be optimal
  - \([-3, 10]\): \(10 \geq \max (0, -9) \rightarrow\) can be optimal

\[\Rightarrow \text{Alternatives } [0, 5] \text{ and } [-3, 10] \text{ can be optimal}\]

**Hurwicz optimism-pessimism criterion**: \((\alpha) \cdot \frac{x_i}{x_j} + (1-\alpha) \cdot \frac{x_j}{x_j}\)

<table>
<thead>
<tr>
<th>(\alpha=0)</th>
<th>(\alpha=0.5)</th>
<th>(\alpha=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>([0, 5])</td>
<td>0</td>
<td>2.5</td>
</tr>
<tr>
<td>([-4, -1])</td>
<td>-4</td>
<td>-2.5</td>
</tr>
<tr>
<td>([-3, 10])</td>
<td>-3</td>
<td>3.5</td>
</tr>
</tbody>
</table>

**Best**: \([0, 5]\; [-3, 10] \; [-3, 10]\)

\((0.5)(-4) + (0.5)(-1) = -2 - 0.5\)

\((0.5)(10) + (0.5)(3) = 5 - 1.5\)
10. Write down:

- an expression for which an optimizing compiler improves the estimates provided by straightforward interval computations, and
- an expression for which an optimizing compiler worsens improves the estimates provided by straightforward interval computations.

These expressions should be different from the examples that I gave; minor difference is OK.

**Improve:**

\[ x^2 + x - 3xy - 3y \quad \Rightarrow \quad (x + 1)(x - 3y) \]

(7 operations) \quad (4 operations)

For \( x = [1, 3], y = [2, 4] \):

- \([1, 3]^2 + [1, 3] - 3[1, 3][2, 4] - 3[2, 4] = ([1, 3] + 1)([1, 3] - 3[2, 4])\)
- \([1, 4] + [1, 3] - 3[2, 12] = [6, 12] = ([2, 4])([1, 3] + [-12, -6])\)
- \([-10, 6] - [6, 36] = [-46, 0] = [2, 4] *[{-11, -3}]\)
- \([-44, -6]\)

**Worsen:**

\[ \frac{1}{a + \frac{1}{b}} \quad \Rightarrow \quad \frac{a \cdot b}{a + b} \]

(9 ops) \quad (3 ops)

\( a = [1, 2], b = [2, 4] \):

- \([1, 2] + \frac{1}{[2, 4]}\)
- \([\frac{1}{[2, 1]} + \frac{1}{[2, 4]} = \frac{[2, 3]}{[1, 2]} = [\frac{3}{2}, \frac{4}{3}]\)
- \([\frac{1}{[2, 1]} + \frac{1}{[2, 4]} = \frac{[2, 3]}{[1, 2]} = [\frac{3}{2}, \frac{4}{3}]\)

\( \frac{[1, 2] \times [2, 4]}{[1, 2] + [2, 4]} = \frac{[2, 8]}{[3, 6]} = \frac{[2, 8]}{[3, 6]} = [\frac{2}{3}, \frac{4}{3}] \)

Width = \(\frac{7}{3}\)
11. If we use 2-digit decimal numbers, what will be the result of multiplying the intervals [0.30, 0.65] and [0.77, 1.30]?

\[
\begin{align*}
1.30 \times 0.65 &= 0.945 \\
1.30 \times 0.3 &= 0.39 \\
0.77 \times 0.65 &= 0.5075 \\
0.77 \times 0.3 &= 0.231 
\end{align*}
\]

Real result = [0.231, 0.945]

2-digit decimal interval (rounded) = [0.23, 0.95]